Universidad Carlos III de Madrid - Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO

TÉCNICAS DE PREDICCIÓN PARA ENERGÍA RENOVABLE

Autor: Diego Mateos Vázquez
Tutor: José María Valls Ferrán
Tutor: Ricardo Aler Mur

Universidad Carlos III de Madrid

2015

Agradecimientos

Alcanzado este punto del mapa, ha llegado la hora de cerrar una etapa. Hace cinco años, tras haber crecido para comprobar que el mundo puede no ser tan maravilloso como muchas personas suelen decirnos cuando somos niños, decidí emprender este viaje, un viaje que en este momento llega a su final y que me ha ayudado a comprender que lo que tenemos delante y detrás de nosotros es poco importante comparado con lo que se encuentra en nuestro interior.

El potencial que hay dentro de cada uno de nosotros tiene una fuerza que no se puede equiparar a ninguna otra cosa, y aguarda en nuestro interior, esperando a que llegue el momento en que nos demos cuenta de que el único impedimento para conseguir las cosas que nos proponemos somos nosotros mismos. Pasamos gran parte de nuestras vidas lamentándonos por las cosas que no podemos conseguir, en lugar de centrarnos en tomar las que tenemos delante e ir tirando de ellas a ver hacia dónde nos llevan. Cuando nos damos cuenta de esto, el abanico de posibilidades a nuestro alcance se abre, mostrándonos caminos que jamás habríamos imaginado que llegaríamos a recorrer.

A medida que comenzamos nuestro viaje, lo primero que debemos hacer es tirar el mapa que hemos comprado o heredado y comenzar a dibujar el nuestro propio. Debemos coger todas las pautas que nos marcan las personas que nos rodean y guardarlas en una caja, porque en algún momento en el que estemos perdidos nos podrán ser de gran utilidad, pero lo que jamás debemos hacer es guiarnos por ellas de principio a fin, pues estaríamos viviendo el viaje de otra persona en lugar del nuestro. A todos y cada uno de nosotros nos esperan vivencias hechas a nuestra medida, decisiones que deberemos tomar por nosotros mismos y no por lo que hagan los demás, pues estas decisiones acabarán marcando en mayor o menor medida quiénes seremos en el futuro.

Si hay algo que he visto claro durante mi viaje, es que cuando llega el momento de elegir entre conciencia y reputación, debemos elegir siempre la primera. Defiende siempre tu trabajo en función de los puntos buenos que crees que tiene, no en función de los fallos del trabajo de los demás o de las razones por las cuales crees que el trabajo de tu compañero es peor que el tuyo. El resto del mundo podrá pensar lo que quiera sobre nuestra manera de proceder, pero si nunca dejamos de ser fieles a nosotros mismos, nunca tendremos que preocuparnos por nada de lo que los demás piensen de nosotros. Debemos tener claro que por cada cosa que hacemos bien, hay otras miles de personas que lo pueden hacer igual o mejor que nosotros, lo cual no es algo malo ni de lejos, sino todo lo contrario. La clave del éxito es comprender que lo que hagamos nosotros solos nunca podrá superar a lo que podamos hacer con la ayuda y colaboración de otros.

Es por eso por lo que lo más preciado son las personas con las que compartimos nuestro viaje. Ni el aprendizaje, ni las notas, ni los reconocimientos, lo más valioso son las personas. Sin ellas, nada habría sido igual a como ha sido. Es por ello por lo que mis agradecimientos se centrarán en esas personas. Por supuesto que estoy agradecido a mi familia y amigos por su apoyo, pero quienes han marcado mi viaje han sido las personas con las que lo he compartido.

Además de llevar a cabo mis estudios, también he tenido el privilegio de trabajar en la universidad. Una experiencia única que jamás habría imaginado antes de iniciar mi viaje, y que me ha llevado a emprender un segundo viaje dentro de la universidad, pero esta vez con un contrato laboral, algo que no habría conseguido sin la ayuda de Juanto y Gloria. Es por lo que mis primeros agradecimientos van para ellos, porque ellos han sido mis verdaderos mentores en este viaje, además de haberse convertido en amigos insustituibles.

Embarcarme en la beca del servicio de audiovisuales de la universidad fue una de las mejores decisiones que tomé, y por ello quiero dar las gracias a todas las personas con las que he compartido esta experiencia. A Iván y Jose, por amenizar mis tardes de lucha con el TFG y por hacer de psicólogos y consejeros en los momentos de bloqueo. A Kike, Yemi, Yoli, Dani, Jan y Nuria, además de todos los mencionados antes, por los grandes momentos vividos dentro y fuera del trabajo y los muchos que nos quedan. A todos los becarios que han sido mis compañeros durante este tiempo: Andrés, Javi, Kamal, Sergio, David, Aris y todos los demás, que han sido muchos y no puedo nombrarlos a todos.

Centrándome en el grado, debo dar las gracias a los que han sido mis mejores compañeros y además amigos fuera de la universidad: Andrés, Carmen, Isaac, Nacho, Pedro, Yemi (que aparece por segunda vez. ¡Cómo no! ¡Siempre metiéndose en todas partes!), Iván y Raquel; porque de cada uno de ellos he aprendido algo diferente, por todas esas prácticas sufridas en grupo, por los momentos surrealistas vividos dentro de las salas de estudio, por todos los ratos que hemos disfrutado juntos, y por los muchos que espero que nos queden. También doy las gracias al resto de compañeros de clase que he tenido y a todos los que han sido en algún momento mis compañeros de prácticas. Porque independientemente de que hayamos trabajado mejor o peor juntos, de todos ellos he aprendido algo que me servirá en el futuro.

Y por último pero no menos importante, debo dar las gracias a mis tutores, Ricardo Aler y José Mª Valls, por toda la ayuda y dedicación que han aportado a este proyecto y por todo lo que me han enseñado.

Siempre, al llegar al final de una etapa de nuestras vidas, sólo quedan los buenos momentos y las personas. Este viaje llega a su final, pero otros muchos me esperan a la vuelta de la esquina, que sin duda también me aportarán más buenos momentos y personas de las que seguir aprendiendo como hasta ahora.

Resumen

Desde hace años, las energías renovables están consideradas como una alternativa a las energías tradicionales y están ganando cada vez más importancia a causa del calentamiento global. Uno de los tipos de energía renovable más destacados es la energía solar, que se obtiene a partir de la radiación electromagnética procedente del sol y que cada año aporta a la Tierra la energía equivalente a varios miles de veces la cantidad de energía que consume el ser humano. El principal problema de la energía solar es que es un recurso intermitente y no determinista, por lo que es necesario que se pueda predecir cuándo se van a producir situaciones que puedan disminuir la radiación solar. El desarrollo de sistemas de predicción de radiación solar ayudaría a incrementar el número de centrales de energía renovable.

El objetivo de este proyecto es estudiar cuáles son las mejores técnicas de predicción para la producción de energía solar. Para ello, hemos utilizado los datos de una competición celebrada en 2013 en la plataforma *Kaggle*. El objetivo de la competición era descubrir qué técnicas de aprendizaje proporcionan las mejores predicciones a corto plazo de la energía solar total diaria acumulada en 98 estaciones de la Mesonet de Oklahoma, utilizando datos meteorológicos recogidos por el *NOAA/ESRL Global Ensemble Forecast System* (GEFS) entre 1994 y 2009.

Los métodos de predicción de datos con los que hemos abordado este problema son el Modelo de Regresión Lineal, las Máquinas de Vectores de Soporte y *Gradient Boosted Regression*. Con estas tres técnicas hemos realizado un total de seis experimento diferentes. En cada experimento, hemos elegido un método de predicción a utilizar, hemos hecho una selección de los parámetros de dicho método, hemos realizado un estudio de parámetros con el que hemos ajustado el valor de cada parámetro, y hemos creado y entrenado diferentes modelos con los que posteriormente hemos realizado predicciones de la energía solar diaria acumulada en cada estación. Para implementar todo esto, hemos utilizado el lenguaje de programación R.

Una vez finalizados los experimentos y recopilados los resultados de todos ellos, los hemos analizado y comparado para poder determinar cuál de las técnicas que hemos estudiado es la más apropiada para abordar el problema de las predicciones de acumulación de energía solar a corto plazo.

Abstract

For years, renewable energies have been considered an alternative to traditional energy sources and their importance is increasing due to global warming. One of the most prominent types of renewable energy is solar energy, that is obtained from the electromagnetic radiation from the sun and that every year brings to Earth the energy equivalent to several thousand times the amount of energy consumed by human being. The main problem is that solar energy is an intermittent and non-deterministic resource, so it is neccesary to be able to predict when it is going to happen a situation that can reduce solar radiation. The development of sunlight prediction systems would help to increase the number of renewable energy plants.

The objective of this project is study which are the best prediction techniques for the production of solar energy. To do this, we used the data from a competition held in 2013 in the Kaggle platform. The aim of the competition was to discover which learning techniques provide the best short-term predictions of total daily solar energy accumulated in 98 stations of the Oklahoma Mesonet, using meteorological data collected by the NOAA / ESRL Global Ensemble Forecast System (GEFS) between 1994 and 2009.

The data prediction methods we used to address this problem are Linear Regression Model, Support Vector Machines and Gradient Boosted Regression. With these three techniques we have accomplished a total of six different experiments. In each experiment, we chose a prediction method to use, we have made a selection of the parameters for this method, we conducted a study of parameters with which we have adjusted the value of each parameter, and we have created and trained different models that we used later to make predictions of daily solar energy accumulated in each station. To implement this, we used the R programming language.

After completing the experiments and compiled the results of all of them, we have analyzed and compared them in order to determine which of the techniques we have studied is the most appropriate technique to address the problem of short-term predictions of solar energy accumulation.

Tabla de Contenido

Índice de Gráficas

Índice de Tablas

Índice de Ilustraciones

Capítulo 1: Introducción.

La energía renovable es aquella que se obtiene de fuentes naturales que no se agotan, ya sea porque son capaces de regenerarse por medio naturales o por la inmensa cantidad de energía que contienen. Además, su impacto ambiental en la emisión de gases de efecto invernadero es nulo. Las energías renovables más destacables son la solar, la eólica, la geotérmica, la hidroeléctrica, la mareomotriz, los biocarburantes y la biomasa[1].

Éste tipo de energía ha constituido una parte muy importante de la energía utilizada por el ser humano desde tiempos remotos, especialmente la energía eólica, la hidráulica y la solar.

Desde los años 70, las energías renovables están consideradas como una alternativa a las energías tradicionales. Esto fue provocado por su disponibilidad garantizada, tanto en el presente como en el futuro (cosa que no ocurre con los combustibles fósiles, que necesitan miles de años para formarse), y por su menor impacto medioambiental[1].

En la actualidad, las energías renovables están ganando importancia a causa del agravamiento del efecto invernadero y el consecuente calentamiento global.

Dentro de las energías renovables, uno de los tipos más destacados es la energía solar. Éste tipo de energía renovable se obtiene a partir de la radiación electromagnética procedente del sol. Cada año, la radiación solar aporta a la Tierra la energía equivalente a varios miles de veces la cantidad de energía que consume el ser humano[2]. Recogiendo la radiación solar de la manera adecuada, puede ser transformada en energía eléctrica o térmica mediante el uso de paneles solares.

El principal problema[3] de la energía solar es que es un recurso intermitente y no determinista. Este problema puede provocar que la red eléctrica sea inestable, por ejemplo, por el paso de nubes o por cualquier otra situación de este tipo. Por ello, es vital que se pueda predecir cuándo se van a producir éstas situaciones que pueden disminuir la radiación solar.

A medida que va creciendo el número de plantas renovables integradas en la red general de distribución eléctrica, la gestión y distribución de la electricidad se vuelven más complejas por la intermitencia característica de este recurso, poniendo así en riesgo la estabilidad del sistema.

Para afrontar este problema, es muy importante el desarrollo de sistemas de predicción de radiación solar. Esto ayudaría a incrementar la cantidad de centrales de energía renovable sin complicar a su vez la gestión de la red eléctrica, por lo que se necesita un importante desarrollo en esta materia.

1.1. Entorno socio-económico.

Analizando el impacto social y económico de las energías renovables, podemos encontrar numerosas ventajas. Según mostraban los datos del "Estudio del Impacto Macroeconómico de las Energías Renovables en España en 2013"[4], que fue elaborado por la Asociación de Empresas de Energías Renovables (APPA) y publicado en diciembre de 2014, las energías renovables generaron ahorros en el sistema eléctrico por valor de 9197 millones de euros.

Si analizamos la energía solar en particular, según informes de Greenpeace[5], la energía solar fotovoltaica podrá suministrar electricidad en 2030 a dos tercios de la población mundial; y según un estudio del Consejo Mundial de Energía publicado en 2007, el 70% de la energía consumida en 2100 será de origen solar.

El precio de las energías renovables no es tan propenso a las fluctuaciones del mercado, como si lo es el del petróleo o el gas natural. Además, los sistemas de energía eólica y solar son capaces de suministrar energía a regiones que están en vías de desarrollo o que son menos accesibles, lo cual puede implicar que dichas regiones no tengan los medios económicos o infraestructurales necesarios para utilizar combustibles fósiles.

La construcción de plantas solares se suele llevar a cabo en comarcas o zonas rurales más o menos desarrolladas. Este tipo de energías impulsan un desarrollo económico producido por las inversiones que conlleva la construcción de las instalaciones requeridas para explotarlas, el empleo que generan y la actividad que promueven en su entorno, como ingeniería, construcción, servicios, industria auxiliar, etc.

Este desarrollo económico supone también un desarrollo social, con dotaciones de infraestructuras comunes, centros de investigación y formación, y unos servicios sociales que sería imposible que se creasen por sí mismos en las zonas donde se suelen instalar estos proyectos de energía solar[6].

Otros impactos que podrían conllevar las energías renovables serían un cambio en las relaciones políticas internacionales provocado por el fin de la dependencia de algunas naciones sobre otras respecto a la energía, una mayor libertad del consumidor para elegir un proveedor doméstico de energía y mejora en la salud de los ciudadanos, ya que dejaríamos de estar expuestos a los desechos peligrosos y a las emisiones asociadas al uso de los combustibles fósiles[7].

1.2. Objetivos del trabajo.

Dado el problema de las predicciones de radiación solar que acabamos de exponer, el principal objetivo de este proyecto es estudiar la aplicación de técnicas de aprendizaje automático para realizar predicciones para la producción de energía solar.

Partiremos de la competición "*AMS 2013-2014 Solar Energy Prediction Contest*"[8] celebrada entre Julio y Noviembre de 2013 en *Kaggle*[9], que es una plataforma web dedicada a las competiciones de predicción de datos, donde las empresas e investigadores publican sus datos para que estadísticos y mineros de datos de todo el mundo compitan por producir los mejores modelos.

El objetivo de la competición era descubrir qué técnicas de aprendizaje proporcionan las mejores predicciones a corto plazo de la energía solar total diaria acumulada en un total de 98 estaciones de la Mesonet de Oklahoma, que sirven como plantas solares en la competición. La Mesonet de Oklahoma[10] es una red mundial de estaciones de monitorización medioambiental diseñada para medir el tamaño y la duración de fenómenos meteorológicos de mesoescala. El término Mesonet es una combinación de las palabras inglesas "*mesoscale*" (mesoescala) y "*network*" (red). En meteorología, un fenómeno de mesoescala es un fenómeno que tiene una duración de entre 1 y 12 horas, una extensión horizontal de entre 1 y 100 kilómetros, o una altura de entre 1 y 10 kilómetros.

Los datos numéricos de predicción meteorológica de entrada para la competición procedían del *NOAA/ESRL Global Ensemble Forecast System* (GEFS)[11], en castellano Sistema Global de Predicción por Conjuntos. Estos datos incluían los 11 miembros del conjunto y los pronósticos de cinco momentos del día: las 12, 15, 18, 21 y 24 horas. Se proporcionaban datos de 1994 a 2007 como datos de entrenamiento, y datos de 2008 y 2009 como conjunto de datos de test. Para la evaluación final de la competición, se utilizaron datos privados más recientes como datos de test.

En la Ilustración 1, se representa la localización de los puntos de donde proceden los datos del GEFS en un mapa de 16x9 puntos, y la localización de las 98 estaciones de la Mesonet de Oklahoma, que como ya he dicho, eran tomadas como plantas solares en la competición. Los puntos de datos del GEFS están representados en el mapa como puntos azules, y las estaciones o plantas solares están representadas como puntos rojos.

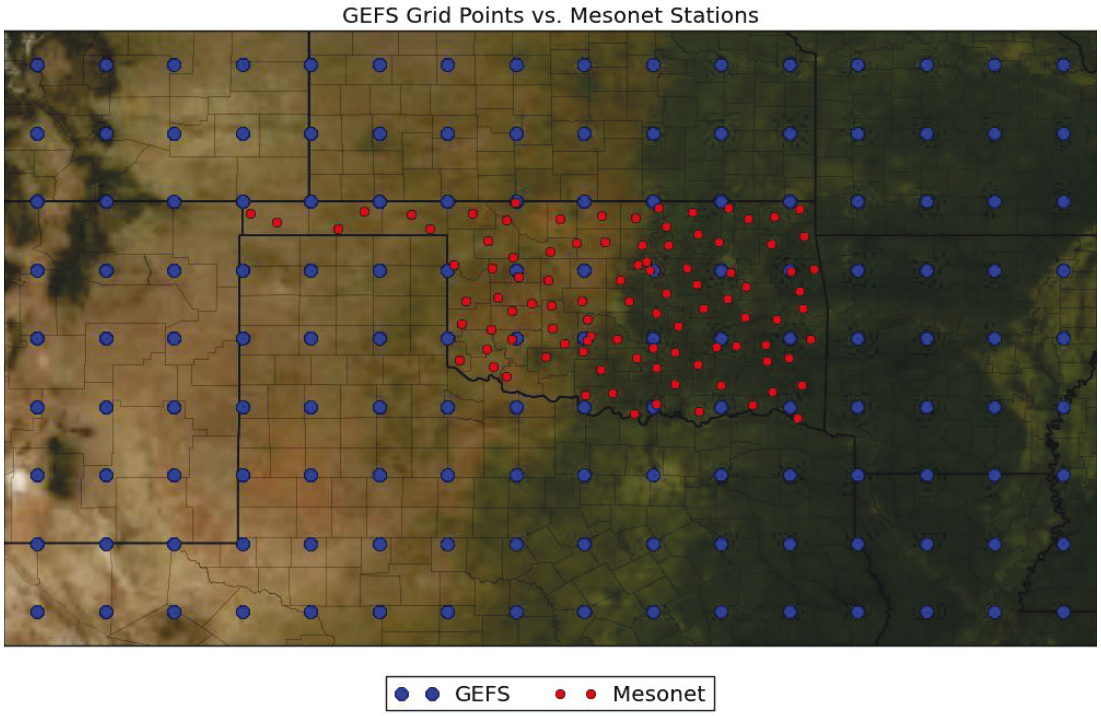


Ilustración 1. Mapa de los puntos del GEFS y las estaciones de la Mesonet de Oklahoma. © 2013 Kaggle, todos los derechos reservados.

Este problema fue abordado previamente en el artículo "*A Study of Machine Learning Techniques for Daily Solar Energy Forecasting using Numerical Weather Models*"[12], publicado por los dos tutores de este Trabajo de Fin de Grado, Ricardo Aler y José M. Valls, junto con Ricardo Martín e Inés M. Galván, todos ellos miembros del Departamento de Informática de la Universidad Carlos III de Madrid, en el área de Ciencia de la Computación e Inteligencia Artificial.

Con este punto de partida, este trabajo será una profundización y extensión de dicho artículo, y además servirá para validar los resultados del mismo. El objetivo principal será estudiar cómo cambia el error de predicción al incrementar el número de puntos del GEFS utilizados para hacer la predicción. Puede parecer lógico que sólo sea relevante el punto más cercano o los 4 que rodean a la estación, pero se quiere comprobar si esto es así y hasta qué punto. Asímismo, se compararán distintos métodos de regresión (Regresión Lineal, Máquinas de Vectores de Soporte (SVM) lineales, polinómicas y gausianas, y *Gradient Boosting Regresion*). Se estudiará si la dependencia del error con respecto al número de puntos del GEFS depende del método y de qué manera. Se medirá también el tiempo de ejecución de cada método, con el objetivo de determinar el coste

computacional que acarrean las mejoras. Como un resultado secundario de todo el proceso de construcción de los modelos, se intentará visualizar cómo depende la precisión de los distintos métodos con respecto a sus parámetros, con el objetivo de comprender la aplicación de dichos métodos.

1.3. Marco regulador.

No existe ninguna regulación, normativa legal ni restricción que se aplique a este problema, ya que todos los datos utilizados son públicos y las herramientas empleadas para llevar a cabo el estudio son software libre.

Los datos que componen el conjunto de datos de entrada que se utilizan en este proyecto son propiedad de *Kaggle*. Todos los derechos reservados.

Los programas utilizados para realizar el estudio, como veremos con más detalle en el apartado 2.2 de este documento, son R y RStudio. R se distribuye con licencia GNU GPL[13] (*General Public License*). La GPL no pone ninguna restricción al uso de R. Tampoco existe ninguna restricción de uso para RStudio, ya que es un IDE (Entorno de Desarrollo Integrado) para R.

Las únicas licencias de uso que ha sido necesario adquirir para realizar este Trabajo de Fin de Grado han sido la del Sistema Operativo y la de Microsoft Office. La adquisición de ambas licencias está reflejada en el presupuesto del proyecto (apartado 6.2 de esta memoria).

1.4. Estructura de la memoria.

En este capítulo, hemos introducido el problema de las predicciones de energía solar y hemos establecido las bases y objetivos a cumplir en el trabajo.

A continuación, pasaremos a analizar los diferentes métodos y herramientas que tenemos a nuestra disposición para abordar el problema de las predicciones, y explicaremos también cómo hemos aplicado dichos métodos a nuestro problema.

Después, explicaremos en detalle los experimentos que hemos realizado. Además, analizaremos los resultados obtenidos en todos ellos y realizaremos las comparaciones pertinentes para concluir eligiendo la mejor o las mejores técnicas de predicción para nuestro problema.

Una vez sacadas dichas conclusiones, estableceremos posibles trabajos futuros que consideramos que podría ser interesante llevar a cabo en base a los resultados que hemos obtenido. Y por último, expondremos la planificación y el presupuesto que se han seguido para realizar este proyecto, así como las referencias y bibliografía consultadas para documentarlo de la manera más precisa posible.

Capítulo 2: Estado del Arte.

En este capítulo, vamos a hablar del tipo de problema que queremos abordar en este estudio, así como de las diferentes técnicas existentes que podemos utilizar para ello y las herramientas que tenemos a nuestra disposición para llevar a cabo dichas técnicas.

El problema que queremos abordar es un problema de regresión, ya que lo que queremos es estudiar diferentes técnicas de predicción para aproximarnos a una salida numérica deseada.

A día de hoy, existen numerosas técnicas para abordar problemas de regresión, y el éxito de muchas de ellas ha ido en aumento en los últimos años debido a los avances que han permitido que se conviertan en técnicas muy fiables, principalmente por la precisión que consiguen al realizar predicciones.

A continuación, nombraremos los diferentes métodos de predicción que vamos a utilizar en nuestro estudio, y explicaremos las principales características del funcionamiento de cada uno de ellos.

Seguidamente, hablaremos también del entorno de programación utilizado para implementar los experimentos que realizaremos con los métodos de predicción elegidos.

2.1. Métodos de predicción utilizados.

De entre todos los métodos de predicción existentes, hemos utilizado los siguientes para nuestro estudio:

2.1.1. Modelo de Regresión Lineal Simple (Linear Model).

La regresión es una técnica estadística que permite representar relaciones entre variables y hacer predicciones con una fiabilidad alta. Un modelo es de regresión cuando, tanto la variable dependiente y las variables independientes, son variables cuantitativas. Modelan la relación entre una variable dependiente *Y*, las variables independientes X_j y un término aleatorio ε. El modelo se puede expresar del siguiente modo[14]:

$Y_t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$, donde:

Y_t: la variable dependiente.

$X_1, X_2, \dots, X_p$: las variables independientes o explicativas.

$\beta_0, \beta_1, \beta_2, \dots, \beta_p$: los parámetros, que miden la influencia que tienen las variables independientes sobre la variable dependiente. β_0 es el término

constante ("intersección"), las $\beta_i (i > 0)$ son los parámetros de cada variable independiente, y p es el número de parámetros independientes que se deben tener en cuenta en la regresión.

La función más simple de regresión es la lineal. Cada variable independiente participa de forma aditiva y constante para todo el dominio observado en la formación de respuesta. El modelo de regresión lineal relaciona la variable dependiente (Y) con *k* variables independientes $\{X_k (k = 1,...K)\}$, o cualquier transformación de éstas variables que generen un hiperplano de parámetros (β_k) desconocidos:

$Y = \sum \beta_k X_k + \varepsilon$, siendo ε el parámetro aleatorio que recoge todos los factores que no se pueden controlar de la realidad (factores asociados con el azar), dándole al modelo un carácter estocástico o no determinista. El caso más sencillo, que es el caso en el cual hay una única variable independiente, el hiperplano resultante es una recta: $Y = \beta_1 + \beta_2 X_2 + \varepsilon$.

El problema de la regresión consiste en elegir unos valores para los diferentes parámetros desconocidos (β_k), para lo que se necesita un conjunto de observaciones. Una observación i-ésima cualquiera ($i = 1,... I$) refleja el comportamiento simultáneo de la variable dependiente y las variables independientes (las perturbaciones aleatorias se suponen no observables): $Y_i = \sum \beta_k X_{ki} + \varepsilon_i$.

Los valores que se eligen como estimadores de los parámetros son los coeficientes de regresión ($\hat{\beta}$). No se puede garantizar que éstos parámetros coincidan con parámetros reales del proceso generador.

Los valores elegidos como estimadores de los parámetros ($\hat{\beta}$) son los coeficientes de regresión, sin que se pueda garantizar que coincidan con parámetros reales del proceso generador. $Y_i = \sum \hat{\beta}_k X_{ki} + \hat{\varepsilon}_i$. Por su parte, los valores $\hat{\varepsilon}_i$ son errores de la perturbación aleatoria.

Si el modelo de regresión lineal cuenta con un número reducido de individuos, muestra una capacidad de generalización mayor que otros modelos más complejos. Esto se debe a la excesiva dependencia de los modelos más complejos respecto de los datos utilizados. El modelo lineal también es más versátil debido a que las variables independientes utilizadas pueden ser transformaciones de las originales.

Hemos elegido este método para nuestro estudio porque es el método más utilizado para predecir valores de variables cuantitativas a partir de los valores de otras variables cuantitativas, y también es el más sencillo.

2.1.2. Máquinas de Vectores de Soporte (Support Vector Machines).

Las Máquinas de Vectores de Soporte (también llamadas Máquinas de Soporte Vectorial o SVM) son consideradas el primer método de la familia de algoritmos denominados métodos kernel, que se han ido desarrollando durante los últimos años. Los métodos kernel se caracterizan por tener como pieza fundamental las llamadas funciones kernel (o simplemente kernel), que son los mecanismos de representación de la información de entrada al algoritmo. Gracias a las funciones kernel, estos métodos son aplicables a todo tipo de datos sin importar a priori la representación original de los mismos, ya que la función kernel adecuada se encarga de transformarlos y efectuar operaciones con dichas transformaciones.

Las SVM fueron introducidas por Vladimir Vapnik a principios de los años noventa a partir de sus trabajos sobre las teorías de aprendizaje estadístico[15]. Desde entonces, han ido ganando reconocimiento debido al buen rendimiento que ofrecen en gran variedad de aplicaciones prácticas.

Lo que diferencia a las SVM de otros métodos de aprendizaje es que estas no se centran en construir hipótesis que cometan pocos errores, sino en producir predicciones en las que se pueda confiar mucho aún a costa de producir ciertos errores, buscando así modelos que estructuralmente tengan poco riesgo de cometer errores ante datos futuros.

Básicamente, a partir de un conjunto de datos de entrenamiento, etiquetamos sus clases y entrenamos una SVM. Así construimos un modelo capaz de predecir la clase de los nuevos datos que se introduzcan.

Antes de poder clasificar, se realiza la etapa de aprendizaje, que consiste en encontrar el hiperplano separador que divida el espacio transformado por la función kernel en dos clases, maximizando la distancia al punto más próximo de cada clase (vectores de soporte), estando así a la misma distancia de los ejemplos más cercanos de cada clase. Para ello se utiliza un conjunto de datos de entrenamiento, al igual que en el resto de algoritmos de aprendizaje supervisado.

En la Ilustración 2 podemos observar la representación de lo explicado anteriormente: Las dos clases (círculos y cuadrados) divididas por el hiperplano separador, maximizando la distancia (margen) entre dicho hiperplano y los puntos más cercanos de cada clase (vectores de soporte).

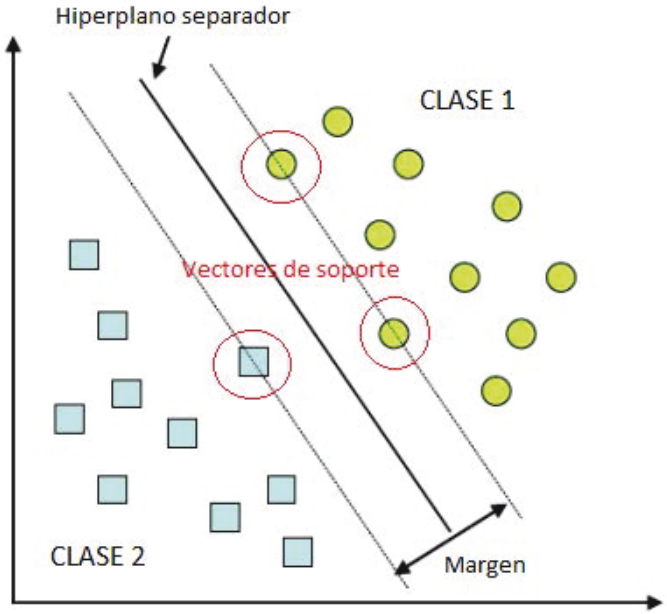


Ilustración 2. Separación de datos mediante SVM.

Tras el aprendizaje, comprobamos el error cometido por el modelo con otra muestra de datos, ya sea conjunto de test o conjunto de validación, comparando la salida obtenida con la salida deseada, que sería la clase real de la muestra.

A partir de aquí, lo que nos quedaría sería calibrar el modelo para que recree la situación deseada con fiabilidad. La calibración consistiría en ajustar la cantidad de datos de entrenamiento. Si no obtuviésemos los resultados deseados, tendríamos que aumentar los datos de entrenamiento hasta conseguir la clasificación correcta. No obstante, al entrenar las SVM el objetivo es hacerlo con el mínimo número de datos de entrenamiento posible, porque a mayor volumen de datos, mayor será el coste de aprendizaje.

Algunas de las funciones kernel más utilizadas en las SVM son:

- Polinómica: $(< x, y > + c)^d$ $c \in \mathrm{R}, \mathrm{d} \in \aleph$
- Gaussiana: $exp\left(\frac{-||x-y||^2}{\gamma}\right)$ $\gamma > 0$
- Sigmoidal: $\tanh(s < x, y > + r)$ $s, r \in \Re$
- Multicuadrática inversa: $\frac{1}{\sqrt{||x-y||^2 + c^2}}$ $c \geq 0$

2.1.3. Gradient Boosted Regression.

Éste método fue originalmente diseñado por Jerome H. Friedman[16][17] para problemas de clasificación, pero también puede ser utilizado en problemas de regresión, mostrando una gran precisión en las predicciones.

Gradient Boosted Regression combina dos algoritmos: los árboles de decisión y el *Boosting*.

Los árboles de decisión son conjuntos de condiciones organizadas jerárquicamente de modo que la decisión final se puede tomar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguno de sus nodos hoja. En su forma más general, las opciones posibles a partir de una condición determinada son excluyentes, por lo que siguiendo el árbol de decisión de manera apropiada, permite llegar a una sola decisión a tomar.

La Ilustración 3 representa un ejemplo de árbol de decisión. En él podemos ver cómo se puede llegar a una acción final, partiendo del nodo raíz hasta llegar a uno de los nodos hoja.

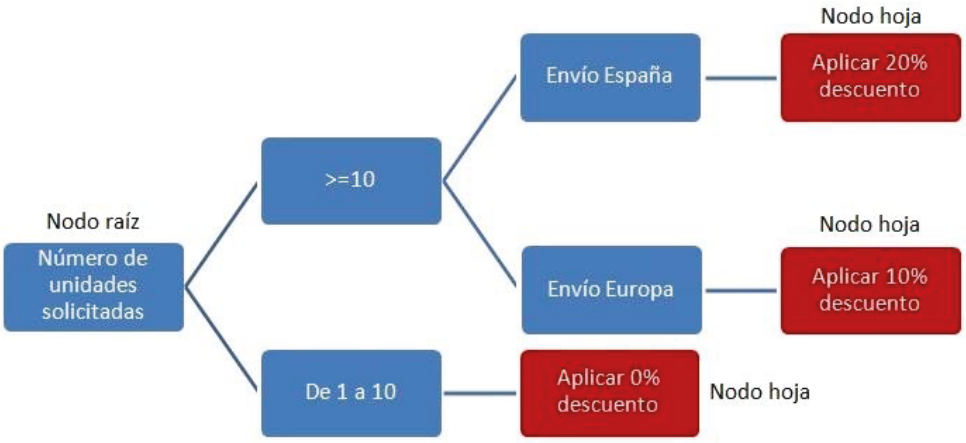


Ilustración 3. Ejemplo de árbol de decisión.

La técnica del *Boosting* consiste en combinar clasificadores (o predictores, en caso de problemas de regresión) que por sí solos no son muy potentes para conseguir un clasificador o un predictor más potente. El aprendizaje del *Boosting* es secuencial: los modelos se van ajustando a los datos de entrada de manera iterativa, poniendo en cada paso más énfasis en las observaciones que no han sido bien ajustadas por los árboles anteriores. Para cumplir este objetivo, los algoritmos de *Boosting* intentan optimizar una función de pérdida, de tal manera que en cada

paso eligen el árbol que reduce más dicha función. El proceso de entrenamiento del modelo es el siguiente [18]:

1. El primer paso es la elección de los parámetros. De este paso depende el éxito del modelo. Los parámetros más básicos son:
    - La complejidad de los árboles, que es el número máximo de nodos por árbol y, por tanto, la complejidad de las interacciones entre las variables.
    - La tasa de aprendizaje, que indica la contribución de cada árbol al modelo final. Si la tasa de aprendizaje es baja, el árbol tardará más en ajustarse pero el resultado será mejor.
    - El número de árboles a ajustar en el modelo. Lo recomendado es utilizar como mínimo 1000 árboles.
    - La proporción de datos que se utilizarán para construir el modelo (*bag fraction*). El resto de datos se usarán para hacer validación cruzada.
2. Una vez elegidos los parámetros, iniciamos el ajuste de los árboles. El primer árbol que se ajusta es el árbol que más reduce la función de pérdida. Para ajustar cada árbol, se utiliza sólo una parte aleatoria de los datos (determinada por el parámetro *bag fraction* que hayamos elegido). Por ello, el proceso de ajuste es estocástico, lo que aumenta la capacidad predictiva del modelo.
3. El segundo árbol a ajustar, en vez de ajustarse a los datos originales, se ajusta a los residuos del primer árbol ajustado. Tras esto, se hacen predicciones para las observaciones teniendo en cuenta los dos árboles ajustados.
4. En los sucesivos pasos, se va ajustando un nuevo árbol sobre los residuos de la combinación de los árboles anteriormente ajustados, por lo que el modelo final es una combinación entre cientos y miles de árboles. La tasa de aprendizaje es crucial, ya que es la que determina la contribución de cada árbol al modelo final, y lo que interesa es que el proceso de construcción del modelo descienda lentamente por la superficie de la función de pérdida para que así el árbol aprenda lento pero seguro.
5. Al finalizar el proceso de ajuste de los árboles, los valores ajustados son la suma de cada uno de los árboles multiplicados por la tasa de aprendizaje.

2.2. Ajuste de parámetros de los métodos de predicción.

Todos los métodos de predicción se componen de una serie de parámetros cuyos valores determinan la precisión de cada uno de ellos. Por ello, llegado el momento de la experimentación, necesitaremos realizar un estudio para ajustar los valores de los parámetros de cada método.

La forma más habitual de ajustar los parámetros es mediante una Búsqueda Exhaustiva en Rejilla o *Grid Search*. Esta técnica consiste en probar todas las combinaciones posibles de los valores de los parámetros. Para cada combinación, se entrena el modelo con un conjunto de entrenamiento pero se calcula el error con un conjunto de validación. Se podría utilizar también validación cruzada, pero supondría un coste computacional mayor. La combinación de valores de los parámetros resultante de la búsqueda será aquella que nos proporcione los mejores resultados, es decir, el error más bajo obtenido con el conjunto de validación.

2.3. Entorno de programación utilizado.

Una vez escogidos los métodos de predicción que vamos a utilizar en nuestro estudio, necesitamos un entorno de programación que nos permita implementar las pruebas que queremos realizar. Dada la cantidad de métodos de predicción que necesitamos implementar y su complejidad, debemos elegir un entorno que tenga disponibles bibliotecas con funciones predefinidas para utilizar todos los métodos que vamos a usar.

Tenemos varias herramientas disponibles[19], pues los métodos que necesitamos son bastante utilizados. Las más destacadas son Weka, Matlab y R. La primera que descartamos fue Weka, porque fue la utilizada en el artículo del que parte este proyecto[12], y queríamos utilizar una herramienta diferente para poder comparar los resultados obtenidos en este estudio y así poder validar los obtenidos en el artículo.

Entre Matlab y R, decidímos utilizar R. Ambas herramientas nos ofrecen características muy similares, y cualquiera de las dos nos habría permitido realizar el estudio con éxito, pero R es software libre y Matlab de pago. Aunque la Universidad cuenta con licencia de Matlab, lo que nos habría permitido utilizar dicha herramienta para realizar este trabajo, decidímos aprovechar que existía una alternativa de software libre y utilizarla. Además, durante el grado utilicé en varias ocasiones Matlab pero no conocía R, por lo que así podía aprender un lenguaje nuevo.

2.3.1. La herramienta R.

R[20] es un entorno y lenguaje de programación orientado a objetos para análisis estadístico y gráfico. Es un proyecto de software libre, resultado de la implementación GNU del lenguaje S. R es el lenguaje más utilizado en investigación estadística, y además cuenta con una popularidad especial en los campos de la investigación biomédica, las matemáticas financieras y la bioinformática.

Fue desarrollado en 1993 por dos miembros del Departamento de Estadística de la Universidad de Auckland, Robert Gentleman y Ross Ihaka. El nombre R viene de la letra inicial del nombre de sus dos creadores. Actualmente, su desarrollo está en manos del llamado *R Development Core Team*[21].

R ofrece una amplia gama de herramientas estadísticas (como modelos lineales, modelos no lineales, análisis de series temporales, algoritmos de clasificación, etc.) y herramientas gráficas para representación de datos.

Las principales ventajas[22] de utilizar R son:

- R se distribuye bajo licencia GNU. Esto quiere decir que su utilización es completamente libre y gratuita, pudiendo incluso mejorarlo.
- R es multiplataforma. No sólo está disponible para los sistemas operativos Windows, GNU/Linux, Unix y Macintosh, también puede integrarse con distintas bases de datos y se puede utilizar desde lenguajes de programación interpretados como Python. Además, se ha desarrollado una interfaz que permite la interacción con la herramienta Weka, que permite leer y escribir ficheros en el formato de dicha plataforma para así poder enriquecer R con sus algoritmos.
- Se puede utilizar cualquier tipo de datos. También es compatible con todos los formatos de datos conocidos (csv, xls, etc.).
- Su capacidad gráfica permite generar gráficos de alta calidad. Cualquier otro paquete estadístico no supera dicha capacidad.
- Se puede extender R mediante paquetes desarrollados por la comunidad de usuarios, ya que es un proyecto completamente abierto y colaborativo, y los usuarios pueden publicar paquetes que extiendan su configuración básica en el repositorio oficial de paquetes[23].

Las primeras pruebas de este proyecto se ejecutaron utilizando la versión 3.1.1 de R. Pero al poco tiempo se hizo pública la versión 3.1.2 (31 de Octubre de 2014), y pasamos a utilizar esta versión.

2.3.2. La interfaz RStudio.

RStudio[24] es un Entorno de Desarrollo Integrado (IDE)[25] para R. Existen dos ediciones de Rstudio: RStudio Desktop, que se ejecuta localmente como una aplicación de escritorio; y RStudio Server, que permite el acceso a RStudio utilizando un navegador web mientras se ejecuta de manera remota en un servidor Linux. RStudio está desarrollado en C++[26], y utiliza el marco Qt[27] para su interfaz gráfica de usuario. La primera versión de RStudio fue publicada en Febrero de 2011.

RStudio está compuesto por cuatro áreas de trabajo diferentes. En la Ilustración 4, podemos ver una muestra de la interfaz donde se aprecian dichas áreas:

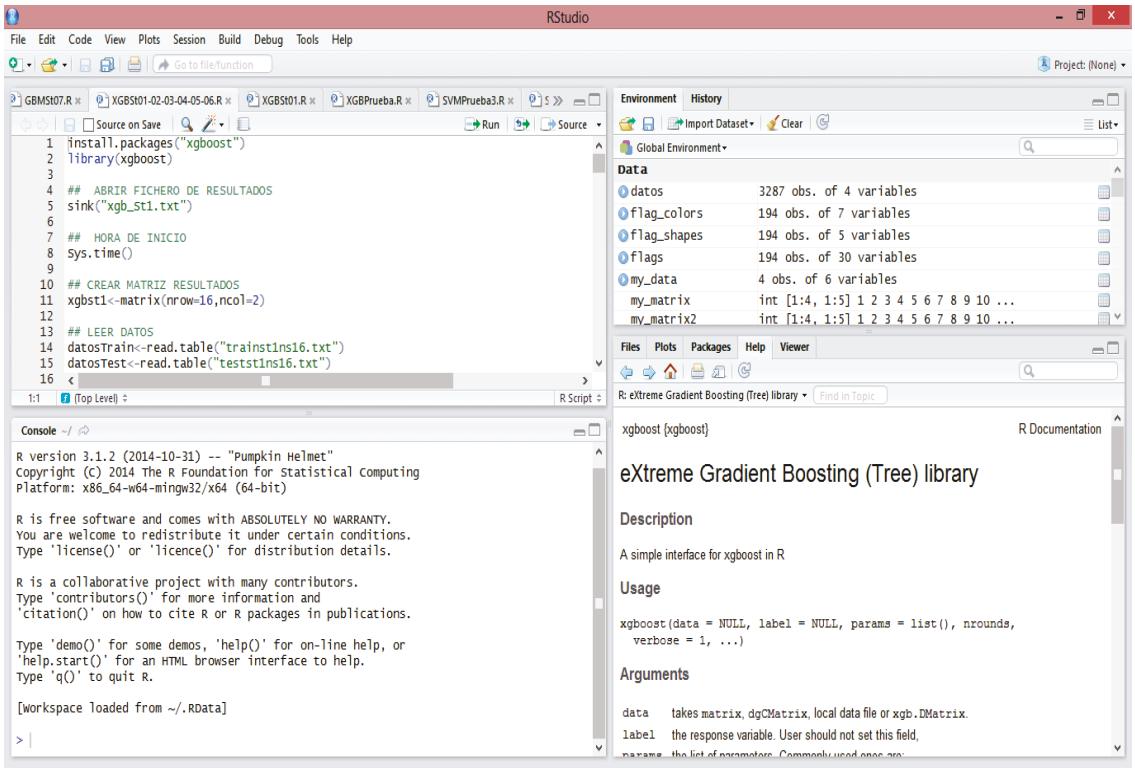


Ilustración 4. Interfaz de RStudio.

En el área superior izquierda se pueden abrir, crear y editar ficheros, generalmente scripts con código R, aunque también se puede trabajar con ficheros de otros tipos.

En el área inferior izquierda, está ubicada la consola de comandos de R. Desde el área superior izquierda también podemos seleccionar partes de scripts o un script entero y enviarlo a ejecutar en esta consola.

El área superior derecha se compone de dos pestañas:

- *Workspace*: Contiene la lista de objetos creados en memoria.
- *History*: Contiene el historial de líneas de código ejecutadas en la consola de R.

Y por último, el área inferior derecha también está compuesta por varias pestañas:

- *Files*: Da acceso al árbol de directorios y ficheros del disco duro.
- *Plots*: En esta pestaña, aparecen los gráficos que se crean en la consola mediante comandos. Tenemos varias opciones, como hacer zoom en los gráficos o exportarlos a un archivo.
- *Packages*: Desde esta pestaña podemos administrar los paquetes de R que están instalados en la máquina. También podemos acceder al repositorio de paquetes de R para instalar los paquetes que necesitemos utilizar y no estén ya instalados.
- *Help*: En esta pestaña tenemos acceso a toda la documentación oficial de R. Si ejecutamos en la consola la ayuda de un comando de R, la página de ayuda de dicho comando se abre aquí.
- *Viewer*: Panel que puede ser utilizado para ver contenido web local. Soporta tanto contenido web estático como apicaciones web locales creadas con Shiny, Rook u OpenCPU. Esto es especialmente útil para paquetes de R que están ligados a bibliotecas de visualización de datos de Javascript.

El aporte técnico de RStudio a R e inexistente, pues únicamente es una interfaz para R. Decidí utilizarlo porque facilita mucho el trabajo a nivel de usuario, especialmente en experimentos relativamente extensos como este, pues permite tener en una misma interfaz todo lo que se puede llegar a necesitar.

La versión de RStudio utilizada en este proyecto es la 0.98.1062. Al igual que R, es software libre. El único requisito necesario para su instalación es haber instalado previamente la versión de R que queramos asociar a RStudio, que como ya he dicho en el apartado anterior, en nuestro caso inicialmente fue la versión 3.1.1 y posteriormente la 3.1.2.

Capítulo 3: Diseño del Sistema.

En este capítulo, explicaremos las decisiones que hemos tomado con respecto al diseño del sistema, que en el problema que estamos abordando sería cómo hemos decidido implementar los métodos de predicción utilizados.

El primer paso que debemos dar para llevar a cabo nuestro estudio, es diseñar el sistema que vamos a utilizar para realizar los diferentes experimentos. En el caso de este proyecto, lo que necesitamos es localizar las funciones necesarias para crear los diferentes modelos de predicción de datos explicados en el capítulo anterior, elegir los parámetros adecuados de dichas funciones para realizar nuestras pruebas y, por supuesto, crear los conjuntos de datos de entrenamiento y test que vamos a utilizar.

3.1. Implementación de los métodos de predicción en R.

En éste apartado, explicaremos detalladamente cómo se utilizan los métodos de predicción explicados en el apartado 2.1, así como los parámetros elegidos en cada uno de los casos.

El entorno de programación R, como ya se ha explicado anteriormente, funciona a base de paquetes que extienden su configuración básica. Para cada uno de los métodos de predicción que hemos utilizado en nuestro estudio, existen varios paquetes que implementan dichos algoritmos. A continuación, explicamos cuáles hemos elegido para llevar a cabo nuestros experimentos y cómo se implementa cada uno de ellos.

3.1.1. Modelo Lineal (LM).

Para ajustar modelos lineales, R tiene disponible la función *lm*[28]. Esta función puede ser utilizada para llevar a cabo regresiones y análisis de varianza y covarianza.

A la función *lm* se le pueden pasar varios parámetros, pero para nuestro experimento sólo utilizaremos el parámetro *formula*, que es la descripción simbólica del modelo a ajustar, y el parámetro *data*, que es el conjunto de datos de entrenamiento que utilizaremos para ajustar el modelo. El resto de parámetros tomarán su valor por omisión.

Una vez creados los modelos, utilizaremos la función *predict*[29] para hacer las predicciones. A ésta función le pasaremos por argumento el modelo previamente creado con *lm* y el conjunto de datos de test.

Estas dos funciones son suficientes para implementar modelos lineales en R para nuestro estudio. Las llamadas a dichas funciones quedarían así:

"modelo <- lm(salida~., data = datostrain)"

"prediccion <- predict(modelo, datostest)"

3.1.2. Máquinas de Vectores de Soporte (KERNLAB).

Existen varios paquetes de R que implementan Máquinas de Soporte Vectorial. Los más conocidos son los paquetes e1071, kernlab, klar y svmpath. Tras investigar cada uno de ellos, decidimos utilizar para nuestros experimentos el paquete kernlab [30], que por las fechas en las que teníamos que elegir era el que había sido actualizado más recientemente.

El paquete kernlab contiene diferentes métodos kernel de aprendizaje automático para clasificación, regresión, clusterización, detección de anomalías, regresión cuadrática y reducción de dimensionalidad.

De este paquete sólo utilizaremos las SVM, y de todas las funciones, las únicas que nos interesan son la función para crear y entrenar los modelos y la función para realizar las predicciones.

La función que proporciona kernlab para crear los modelos se llama *ksvm* [31]. *Ksvm* utiliza el algoritmo SMO de John Platt [32] para resolver el problema QP de las SVM y la mayoría de las formulaciones de las SVM. Los datos se le pueden pasar tanto en forma de matriz como en un *data.frame*, e incluso en forma de una matriz kernel de tipo *kernelMatrix* o como una lista de vectores de tipo carácter cuando se utiliza el kernel string.

Los parámetros de la función *ksvm* que necesitamos son:

- *kernel*: Éste parámetro indica la función kernel utilizada para entrenar el modelo y hacer las predicciones. Kernlab ofrece las funciones kernel más populares, y se definen de la siguiente manera:
  - "*rbfdot*": Kernel Gaussiano (*Radial Basis Kernel*).
  - "*polydot*": Kernel polinómico.
  - "*vanilladot*": Kernel lineal.
  - "*tanhdot*": Kernel sigmoidal.
  - "*laplacedot*": Kernel de Laplace.
  - "*besseldot*": *Bessel kernel*.
  - "*anovadot*": *ANOVA RBF kernel*.
  - "*splinedot*": *Spline kernel*.

- "*stringdot*": Kernel String.

- *C*: Éste parámetro es el coste por violación de restricciones. Es la constante C del término regulador en la fórmula de Lagrange. Por defecto, su valor es 1.
- *kpar*: Es la lista de parámetros del kernel. Contiene los parámetros que se usan con la función kernel elegida. Los parámetros válidos para cada función kernel implementada para *ksvm* son:
  - *sigma*: para kernel Gaussiano y de Laplace.
  - *degree*, *scale* y *offset*: para el kernel polinómico.
  - *scale* y *offset*: para kernel sigmoidal.
  - *sigma*, *order* y *degree*: para *Bessel kernel*.
  - *sigma* y *degree*: para kernel ANOVA.
  - *lenght*, *lambda* y *normalized*: para kernel string.

  En caso del kernel Gaussiano, también se puede poner el valor "*automatic*", que utiliza la heurística implementada en la función de kernlab llamada *sigest* para calcular un buen valor de sigma.
- *epsilon*: Como su nombre indica, es el valor de la variable epsilon (margen de tolerancia).

Además de estos parámetros, debemos pasarle a la función la fórmula o descripción simbólica del modelo a ajustar y, por supuesto, el conjunto de datos de entrenamiento.

Una vez entrenado el modelo, debemos utilizar la función *predict*[33] para realizar las predicciones. A esta función únicamente necesitamos pasarle el modelo creado con la función *ksvm* y el conjunto de datos que queramos utilizar para predecir (el conjunto de datos de test, aunque nosotros en este estudio hemos hecho predicciones tanto con el conjunto de datos de entrenamiento como con el de test).

Como las SVM ofrecen tantas posibilidades, nos pareció interesante probar varias alternativas, así que decidimos realizar tres experimentos con ellas con tres kernels diferentes: SVM con kernel Gaussiano (*kernel="rbfdot"*), SVM con kernel polinómico (*kernel="polydot"*) y SVM con kernel sigmoidal (*kernel="tanhdot"*).

Como he explicado anteriormente, cada función kernel tiene sus propios parámetros. Los parámetros *C* y *epsilon* son comunes a los tres kernels, pero de entre los parámetros propios de cada uno hemos utilizado:

- *rbfdot*: Para el experimento con kernel Gaussiano, utilizaremos el parámetro *sigma*, que controla la anchura de la campana de Gauss. Con valores grandes de sigma, los puntos de nuestras características conocidas podrán tomar más valores en su dominio, teniendo así cambios más suaves en sus valores. Por el contrario, con valores pequeños tendremos un dominio de valores más reducido, por lo que tendremos más varianza.
- *polydot*: De los tres parámetros que tiene, sólo utilizaremos *degree*, que indica el grado del polinomio. Únicamente probaremos con grado 2, porque grado 1 equivale a un modelo de regresión lineal, y grados superiores a 3 podrían necesitar mucho tiempo de cómputo.
- *tanhdot*: En este caso, utilizaremos los valores por defecto por omisión de parámetros, porque lo que nos interesa es ver qué tipo de valores resultantes obtenemos, pero no creemos que merezca la pena realizar el experimento completo porque los resultados no deberían ser muy buenos en comparación con los otros kernels.

Con los parámetros elegidos, las llamadas a las funciones serían del tipo:

"modelo <- ksvm(salida ~., datostrain, kernel = "rbfdot", C = 1.0, kpar = list(sigma = 0.1), epsilon = 0.1)"

"modelo <- ksvm(salida ~., datostrain, kernel = "polydot", C = 1.0, kpar = list(degree = 2), epsilon = 0.1)"

"modelo <- ksvm(salida ~., datostrain, kernel = "tanhdot", C = 1.0, epsilon = 0.1)"

"prediccion <- predict(modelo,datostest)"

3.1.3. Generalized Boosted Regression Models (GBM).

Existen diversos paquetes que implementan modelos de *boosting*. Uno de los más utilizados es GBM[34]. Este paquete implementa extensiones del algoritmo AdaBoost de Freund y Schapire[35] y de la *Gradient Boosting Machine* de Jerome H. Friedman[16]. Incluye también métodos de regresión para mínimos cuadrados, logística y Poisson, entre otros muchos.

Para ajustar los modelos, tenemos la función *gbm*[36], que se llama igual que el paquete. Los parámetros que vamos a necesitar en esta función son:

- *n.trees*: Este parámetro indica el número total de árboles que queremos ajustar en el modelo. Esto equivale al número de iteraciones y de funciones base en la expansión.
- *shrinkage*: Es la tasa de aprendizaje a aplicar a los árboles.
- *interaction.depth*: Es el parámetro que indica la profundidad máxima de los árboles. Un valor de 1 implica un modelo aditivo, un valor de 2 implicaría un modelo con un máximo de dos caminos/interacciones, etc.
- *distribution*: Éste parámetro sirve para indicar la distribución a usar. Las opciones disponibles son: "*gaussian*" (error cuadrático), "*laplace*" (pérdida absoluta), "*tdist*" (distribución t de Student), "*bernoulli*" (regresión logística para salidas 0-1), "*huberized*", "*multinomial*" (clasificación cuando hay más de dos clases), "*adaboost*" (pérdida exponencial de AdaBoost para salidas 0-1), "*poisson*", "*coxph*", "*quantile*" y "*pairwise*" (ranking de medida utilizando el algoritmo LambdaMart). Si no se especifica el valor de este parámetro, *gbm* intenta averiguar el apropiado: si la respuesta tiene únicamente dos posibles valores, se asume el valor "*bernoulli*"; si no, si la respuesta es un factor, se asume "*multinomial*"; si no, si la respuesta es de la clase "*Surv*" (*Survival object*), se asume "*coxph*"; y si no, se asume "*gaussian*".

También hay que pasarle, como en los casos anteriores, el conjunto de datos de entrenamiento y la fórmula del modelo a ajustar.

En cuanto a la función de predicción, además de pasarle el modelo entrenado y el conjunto de datos de test, también debemos pasarle el número de árboles que debe utilizar para hacer las predicciones. Lo habitual es que éste parámetro tenga el mismo valor que el parámetro "*n.trees*" de la función *gbm*, pero también puede tener un valor menor. De este modo, si por ejemplo tenemos un modelo ajustado con 5000 árboles y a la función *predict*[37] le decimos que utilice 3000 árboles, aunque el modelo esté creado con 5000 árboles, las predicciones se harán utilizando únicamente 3000.

Con los parámetros elegidos para este caso, las llamadas a las funciones para crear los modelos y realizar las predicciones serán de este estilo:

"modelo <- gbm(salida~., data = datosTrain, n.trees = 5000, shrinkage = 0.01, interaction.depth = 10, distribution = "laplace")"

"prediccion <- predict(modelo, datosTrain, 5000)"

3.1.4. eXtreme Gradient Boosting Training (XGBoost).

Debido al auge de las técnicas de *boosting* y a su éxito y precisión al aplicarlas en este tipo de problemas, decidimos utilizar un segundo paquete para poder comparar su funcionamiento con el paquete anteriormente explicado (GBM).

XGBoost[38] es una biblioteca optimizada de técnicas de *Gradient Boosting*. Dicha biblioteca está paralelizada usando *OpenMP*[39], que es una interfaz de programación de aplicaciones para la programación multiproceso de memoria compartida en múltiples plataformas. Implementa un algoritmo de aprendizaje automático bajo técnicas de *Gradient Boosting*, incluyendo el modelo lineal generalizado y regresión de árbol de decisión impulsado. Esta biblioteca está adaptada a R con un paquete homónimo.

El paquete XGBoost[40] para R implementa varias funciones objetivo, incluyendo regresión, clasificación y ranking. Además, el paquete está diseñado para ser extensible, por lo que los usuarios pueden definir sus propias funciones objetivo de manera sencilla.

Decidimos incluir este paquete en el estudio porque ha sido utilizado para ganar múltiples competiciones de *Kaggle* y por las ventajas que ofrece respecto al paquete GBM clásico:

- Velocidad: Puede computar de manera paralelizada automáticamente tanto en Windows como en Linux, por lo que es hasta diez veces más rápido que gbm.
- Dispersidad (*Sparsity*): Acepta entradas dispersas tanto para el *booster* de árboles como para el lineal.
- Personalización: Soporta funciones objetivo y de evaluación personalizadas.

Para crear los modelos, nos proporciona la función *xgboost*[41]. Los parámetros son diferentes a los utilizados en la función *gbm*.

- *data*: Es el conjunto de datos de entrada. Las diferencias con respecto a *gbm* es que hay que utilizar matrices en lugar de *data.frames* y que hay que separar las entradas de las salidas.
- *label*: La variable de respuesta o salida.
- *max.depth*: Es equivalente al parámetro "*interaction.depth*" de la función *gbm*. La profundidad máxima de los árboles.

- *eta*: Es la tasa de aprendizaje. Equivalente al parámetro *shrinkage* de *gbm*.
- *nrounds*: Número máximo de iteraciones o árboles. Equivalente a *n.trees* en la función *gbm*.
- *objective*: Este parámetro sirve para indicar la función objetivo. Los valores más comunes son "*reg:linear*" (regresión lineal) y "*binary.logistic*" (regresión logística para clasificación).

En cuanto a la función *predict*[42] para hacer la predicciones, los parámetros son los mismos que hemos utilizado en GBM.

De esta manera, las llamadas a las funciones que utilizaremos para implementar los modelos y predicciones tendrán la siguiente forma:

"modelo <- xgboost(data = entrada, label = salida, max.depth = 6, eta = 0.1, nround = 10000, objective = "reg:linear")"

"pred1train<-predict(modelo, datos, ntreelimit = 10000)"

Una vez elegidos los paquetes y estudiadas las funciones que necesitamos para realizar nuestro estudio, el siguiente paso es realizar el estudio de parámetros para cada caso y determinar así los valores de los parámetros de cada función para realizar los experimentos. Esto se explica detalladamente en el capítulo 4 de este documento.

3.2. Descripción de los datos.

El objetivo es predecir la energía solar diaria acumulada medida en J x m^{-2} en 98 estaciones de la red Mesonet de Oklahoma, que abarca aproximadamente 180000 kilómetros cuadrados (representado en la Ilustración 1 del apartado 1.2 de este documento).

En los datos proporcionados por *Kaggle* para la competición, los datos de entrada para cada día se corresponden con las salidas del modelo numérico de predicción meteorológica del GEFS, utilizando 11 miembros del conjunto y 5 previsiones de entre las 12 y las 24 horas, una previsión cada 3 horas. Cada miembro del conjunto produce salidas para 15 variables meteorológicas por cada una de las 5 previsiones diarias y para cada punto del GEFS que cubren la superficie, incluyendo el Estado de Oklahoma y áreas circundantes. La distancia entre cada punto del GEFS es aproximadamente 90 kilómetros. Las 15 variables meteorológicas recogidas son las siguientes:

VARIABLE	DESCRIPCIÓN	UNIDAD DE MEDIDA
apcp_sfc	Precipitación acumulada en la superficie.	kg m^{-2}
dlwrf_sfc	Flujo de radiación descendente de onda larga promedio en la superficie.	W m^{-2}
dswrf_sfc	Flujo de radiación descendente de onda corta promedio en la superficie.	W m^{-2}
pres_msl	Presión atmosférica media al nivel del mar.	Pa
pwat_eatm	Agua de precipitación en toda la superficie de la atmósfera.	kg m^{-2}
spfh_2m	Humedad específica a 2 metros por encima de suelo.	kg kg^{-1}
tcdc_eatm	Cubierta total de nubes en toda la superficie de la atmósfera.	%
tcolc_eatm	Condensación total en la superficie de la atmósfera.	kg m^{-2}
tmax_2m	Temperatura máxima durante las últimas 3 horas a 2 metros por encima del suelo.	K
tmin_2m	Temperatura mínima durante las últimas 3 horas a 2 metros por encima del suelo.	K
tmp_2m	Temperatura actual a 2 metros por encima del suelo.	K
tmp_sfc	Temperatura en la superficie.	K
ulwrf_sfc	Radiación ascendente de onda larga en la superficie.	W m^{-2}
ulwrf_tatm	Radiación ascendente de onda larga en la parte superior de la atmósfera.	W m^{-2}
uswrf_sfc	Radiación ascendente de onda corta en la superficie.	W m^{-2}

Tabla 1. Descripción de las 15 variables meteorológicas de entrada.

De este modo, el número total de atributos de cada punto del GEFS son 11 miembros del conjunto, multiplicados por 5 previsiones (una cada 3 horas entre las 12 y las 24 horas), multiplicado por las 15 variables meteorológicas. 11x5x15 hacen un total de 825 atributos. Como tenemos un mapa de 16x9=144 puntos, el total de datos disponibles para cada día asciende a 118800.

Estos datos fueron recogidos cada día desde 1994 hasta 2007, lo que hace un total de 5113 días. Además, se asoció a cada día la correspondiente radiación solar acumulada, que es el atributo que necesitamos predecir. Esta

radiación solar acumulada se calculó sumando la radiación medida por un piranómetro en cada estación cada 5 minutos desde la salida del sol hasta las 23:55 horas del día correspondiente.

3.2.1. Creación de los conjuntos de datos de entrenamiento y test.

En principio, los puntos del GEFS más cercanos a cada estación deberían ser los más relevantes para realizar las predicciones, pero también queremos estudiar cómo influyen los datos en las predicciones cuando se va aumentando el número de puntos del GEFS como datos de entrada para cada método de predicción. Para ello, se decidió recopilar para cada estación los datos de los 16 puntos del GEFS más cercanos a ella.

Debido al gran volumen de datos que tenemos que manejar para este estudio, se tomó la decisión de reducir la cantidad de atributos de cada punto del GEFS. Cada punto se componía de 825 atributos (11 miembros del conjunto x 5 previsiones cada 3 horas x 15 variables meteorológicas). Lo que se hizo fue utilizar la media de los 11 miembros del conjunto en lugar de utilizarlos todos. De esta manera, la cantidad de atributos por punto se redujo a 5x15=75 atributos.

De la cantidad de datos disponible (14 años), se decidió usar los datos de los primeros 12 años, de 1994 a 2005, como conjunto de datos de entrenamiento (12 años x 365 días = 4380 días) y los datos de los 2 años restantes, 2006 y 2007, como conjunto de test (5113 días totales - 4380 días de entrenamiento = 733 días de test).

Tras tomar estas decisiones, creamos un fichero de datos de entrenamiento y otro de test para cada una de las 98 estaciones de la Mesonet. De cada estación, necesitamos los 75 atributos (15 variables meteorológicas de cada uno de los 5 momentos de las previsiones) de los 16 puntos del GEFS más cercanos a la estación y la radiación solar calculada para cada día en dicha estación.

Por tanto, cada fichero, tanto de entrenamiento como de test, estará compuesto por 1201 columnas (15x5x16+1). El número de filas de cada fichero está determinado por el número de días utilizados. Como ya se ha explicado anteriormente en este mismo apartado, cada fichero de entrenamiento tendrá 4380 filas, correspondientes a los 4380 días que hemos seleccionado como conjunto de datos de entrenamiento. Del mismo modo, los ficheros de datos de test estarán compuestos por 733 filas. Tendremos un total de 196 ficheros de datos de entrada, 98 ficheros de entrenamiento y 98 de test.

Capítulo 4: Experimentación.

En este nuevo capítulo, explicaremos detalladamente todos los experimentos que han compuesto nuestro estudio desarrollado en este proyecto. También analizaremos y compararemos los resultados obtenidos para poder sacar conclusiones sobre el problema que hemos abordado y ver si se hemos podido cumplir los objetivos que nos habíamos marcado.

Cada experimento consta de un conjunto de pruebas por cada estación, utilizando uno de los métodos de predicción explicados anteriormente. Los resultados que utilizaremos para analizar la eficacia de los métodos de predicción es el Error Absoluto Medio (MAE), por tanto, obtener dicho error será el objetivo de nuestras pruebas. Este error se calcula haciendo la media aritmética en valor absoluto de la resta de las salidas producidas por los modelos que hemos entrenado menos los valores de las salidas deseadas (la radiación solar acumulada en cada estación, o lo que es lo mismo, la última columna de nuestros ficheros de datos): $MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i| = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|$, donde $|e_i|$ es el error absoluto, que se calcula $|f_i - y_i|$, siendo f_i la predicción e y_i la salida real.

Como se explicó en el apartado 3.2.1, en los conjuntos de datos de entrada tenemos disponibles, para cada estación, los atributos de los 16 puntos más cercanos a cada una de ellas. Algo que tienen en común todos los experimentos es que, para cada estación, se crean modelos de predicción empezando por utilizar el punto más cercano a cada estación, y se van incrementando los puntos utilizados hasta llegar a un modelo en el que se utilizan los 16 puntos más cercanos. De este modo, no sólo vemos con qué cantidad de puntos se obtienen los mejores resultados, también vemos cómo influye la cantidad de puntos en la evolución del error.

Para lanzar las pruebas que componen nuestros experimentos, necesitamos determinar valores para los parámetros de cada uno de los métodos de predicción. Para obtener dicho valores, llevaremos a cabo un estudio de parámetros por cada método. El estudio de parámetros consistirá en una búsqueda exhaustiva en rejilla con conjunto de entrenamiento y validación para cada combinación de parámetros. Para esta búsqueda necesitaremos crear unos conjuntos de entrenamiento y validación específicos a partir de los datos de entrada que disponemos. A continuación, explicaremos cómo hemos creado dichos conjuntos de entrenamiento y validación, y después pasaremos a explicar los experimentos realizados.

4.1. Creación de los conjuntos de datos de entrenamiento y validación para los estudios de parámetros.

Los conjuntos de datos que utilizaremos para realizar los estudios de parámetros serán mucho más sencillos que los conjuntos utilizados para los experimentos.

Partiremos del conjunto de datos de entrenamiento de la estación número 1 de las que disponemos, ya que para ajustar los parámetros optaremos por utilizar únicamente una de las estaciones, y asumiremos que los parámetros encontrados para la primera estación son válidos para el resto. Esta es una aproximación razonable y reduce el tiempo de cómputo. Dicho conjunto se compone de los datos de 4380 días (12 años). De esos 12 años, utilizaremos 10 años como conjunto de datos de entrenamiento y 2 años como conjunto de datos de validación. De este modo, para el ajuste de parámetros tendremos 3650 días de entrenamiento y 730 de validación.

Reduciremos también la cantidad de atributos de los conjuntos, ya que utilizar los 16 puntos más cercanos a la estación para ajustar los valores de los parámetros sería excesivo. Utilizaremos únicamente los 5 puntos del GEFS más cercanos a la estación.

Así, el conjunto de datos de entrenamiento para el ajuste de parámetros estará compuesto por 376 columnas (75 atributos x 5 puntos + 1 salida) y 3650 filas (365 días x 10 años), y el conjunto de datos de validación se compondrá del mismo número de columnas y 730 filas (365 días x 2 años).

4.2. Descripción de los experimentos realizados.

En este apartado, se describirán los experimentos realizados con cada uno de los métodos de predicción. Por cada experimento, se explicará cómo se realizó el ajuste de sus parámetros y los problemas que fueron surgiendo durante la experimentación. También se expondrán los tiempos de ejecución de los experimentos y los mejores resultados obtenidos.

4.2.1. Experimento 1: LM

En el primer experimento realizado, utilizamos el Modelo de Regresión Lineal. Como se explica en el apartado 3.1.1, la función utilizada para crear los modelos en este experimento en R es *lm*. A dicha función le pasamos la fórmula a ajustar (*salida~.*) y los datos de entrenamiento en un *data.frame*. Por cada una de las 98 estaciones, creamos un total de 16 modelos. Para el primero, utilizamos las columnas de la 1 a la 75 (los 75 atributos del punto más cercano) del fichero de entrenamiento más la 1201 (la salida deseada); para el segundo modelo, utilizaremos de la 1 a

la 150 más la 1201, y así sucesivamente hasta llegar a utilizar todas las columnas del fichero de datos de entrenamiento en el modelo con los 16 puntos más cercanos a la estación.

En cada caso, tras crear los modelos, realizaremos las predicciones de entrenamiento y test. Para ello, utilizaremos la función *predict*, a la que le pasaremos el modelo entrenado y el *data.frame* con los datos de entrenamiento para la predicción de entrenamiento y los de test para la predicción con nuevos datos no utilizados para entrenar el modelo. Los datos para las predicciones se tomarán para el *data.frame* del mismo modo que al entrenar el modelo: incrementando el número de columnas tomadas de 75 en 75 en función del número de puntos del GEFS más cercanos a la estación que estemos utilizando.

Una vez hechas las predicciones, calcularemos el Error Absoluto Medio, tanto de entrenamiento como de test, para cada caso. Estos serán los valores con los que trabajaremos en la parte de análisis de resultados.

Este experimento en su totalidad se compondrá de 1568 modelos (16 por cada estación) y 3136 predicciones (16 de entrenamiento y 16 de test por cada una de las 98 estaciones).

4.2.1.1. Estudio de parámetros.

Como determinamos en el apartado 3.1.1 de esta memoria, en este experimento hemos omitido los parámetros en la llamada a la función para que tomen sus valores por omisión, por lo que no fue necesario llevar acabo ningún ajuste de parámetros, sobre todo porque la regresión lineal no tiene parámetros importantes que ajustar.

4.2.1.2. Tiempos de ejecución.

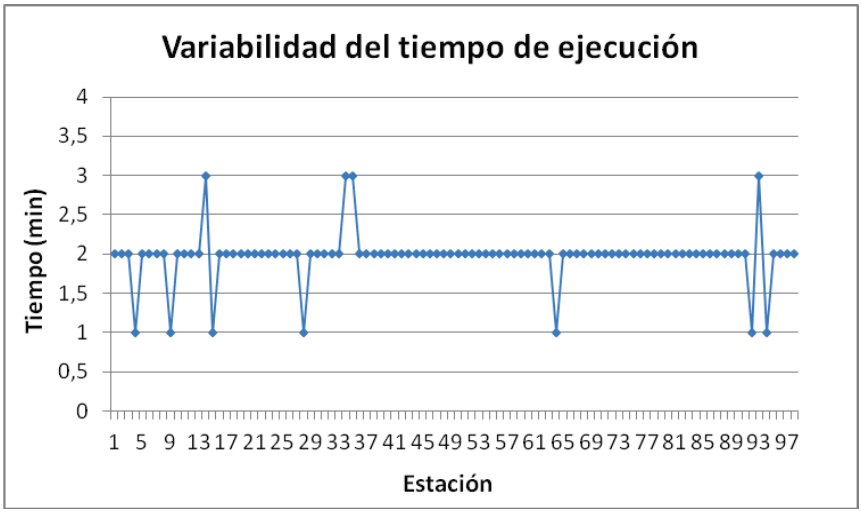
Durante la experimentación, se han calculado los tiempos de ejecución de las pruebas de las estaciones. Dichos tiempos de ejecución incluyen todas las pruebas de cada estación, es decir, el tiempo que ha tardado R en entrenar los 16 modelos y en calcular las predicciones asociadas a cada uno de esos modelos.

En la Tabla 2, se muestran los tiempos de las 98 estaciones medidos en minutos.

Estación	Tiempo	Estación	Tiempo	Estación	Tiempo	Estación	Tiempo
St01	2	**St26**	2	**St51**	2	**St76**	2
St02	2	**St27**	2	**St52**	2	**St77**	2
St03	2	**St28**	1	**St53**	2	**St78**	2
St04	1	**St29**	2	**St54**	2	**St79**	2
St05	2	**St30**	2	**St55**	2	**St80**	2
St06	2	**St31**	2	**St56**	2	**St81**	2
St07	2	**St32**	2	**St57**	2	**St82**	2
St08	2	**St33**	2	**St58**	2	**St83**	2
St09	1	**St34**	3	**St59**	2	**St84**	2
St10	2	**St35**	3	**St60**	2	**St85**	2
St11	2	**St36**	2	**St61**	2	**St86**	2
St12	2	**St37**	2	**St62**	2	**St87**	2
St13	2	**St38**	2	**St63**	2	**St88**	2
St14	3	**St39**	2	**St64**	1	**St89**	2
St15	1	**St40**	2	**St65**	2	**St90**	2
St16	2	**St41**	2	**St66**	2	**St91**	2
St17	2	**St42**	2	**St67**	2	**St92**	1
St18	2	**St43**	2	**St68**	2	**St93**	3
St19	2	**St44**	2	**St69**	2	**St94**	1
St20	2	**St45**	2	**St70**	2	**St95**	2
St21	2	**St46**	2	**St71**	2	**St96**	2
St22	2	**St47**	2	**St72**	2	**St97**	2
St23	2	**St48**	2	**St73**	2	**St98**	2
St24	2	**St49**	2	**St74**	2	**Media**	1,969
St25	2	**St50**	2	**St75**	2	**Desv. Típica**	0,335

Tabla 2. Tiempos de ejecución en minutos del experimento con LM.

Como podemos ver en la tabla, los tiempos de ejecución han sido muy bajos. El tiempo medio de ejecución de este método ha sido 2 minutos aproximadamente. No se observan variaciones significativas entre los tiempos, el mínimo es 1 y el máximo 3, por lo que no hay nada que destacar en ese sentido. En la Gráfica 1 se representa la variabilidad del tiempo de ejecución entre estaciones.



Gráfica 1. Variabilidad del tiempo de ejecución con LM.

4.2.1.3. Problemas surgidos durante la experimentación.

Durante este experimento no han surgido problemas significativos que hayan dificultado la realización del mismo. Todas las pruebas se pudieron realizar correctamente y en un tiempo de ejecución bastante rápido y sin requerir más de un equipo.

4.2.1.4. Mejor resultado para cada estación.

El experimento ha concluido con 3136 resultados de MAE obtenidos. Como son muchos resultados, hemos decidido incluir en éste apartado únicamente los resultados del caso con el que se obtiene el menor MAE de test en cada estación. En la Tabla 3 se muestran dichos resultados, reflejando el número de estación, la cantidad de puntos cercanos que se han utilizado para obtener esos resultados, y los MAE de entrenamiento y test del caso.

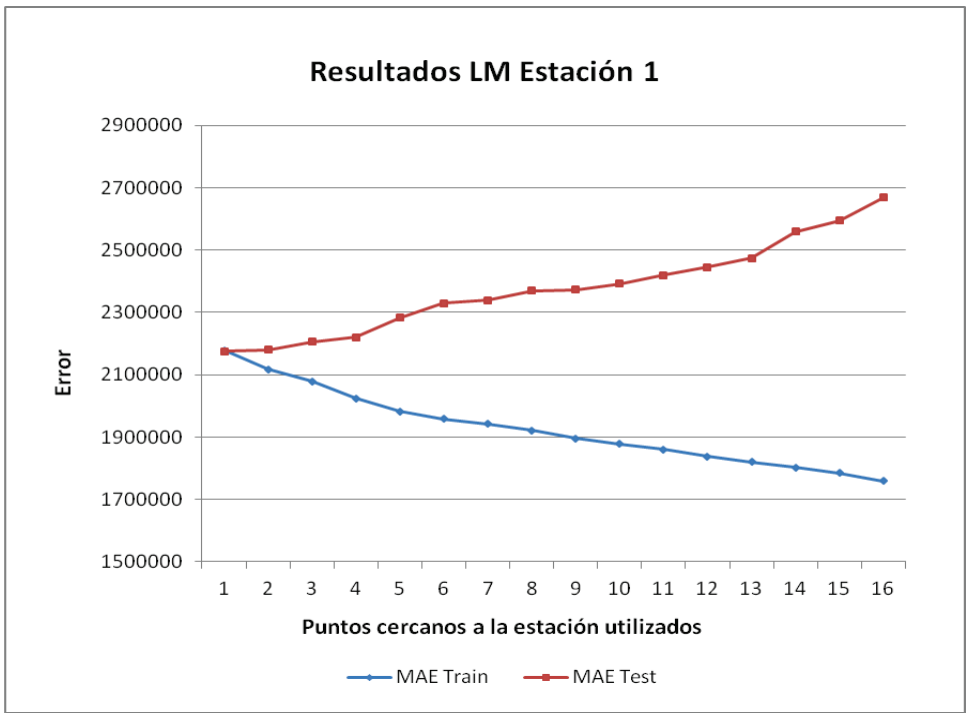
Estación	Puntos	MAE Train	MAE Test	Estación	Puntos	MAE Train	MAE Test
St01	1	2177816	2174914	**St50**	1	2348176	2263141
St02	3	2161959	2091548	**St51**	1	2286641	1944520
St03	2	2169278	2597714	**St52**	1	1998519	2082204
St04	1	2140408	2105064	**St53**	1	2265647	2159403
St05	5	1969344	2057792	**St54**	1	2279845	2290347
St06	2	2054020	2365687	**St55**	4	2077475	2139779
St07	2	2047136	1986859	**St56**	3	2131721	2101835
St08	2	2255684	2162393	**St57**	3	2036025	2169326
St09	1	2342077	2072419	**St58**	3	2163612	2109722
St10	2	1871406	1815350	**St59**	3	2026794	2016868

St11	3	2125076	2115012	**St60**	2	2457954	2580925
St12	3	2069507	2063557	**St61**	1	2174468	2216396
St13	3	2140948	2057608	**St62**	1	2306553	2222360
St14	2	2176629	2266944	**St63**	4	2173420	2167809
St15	2	2246803	2060571	**St64**	2	2134529	2087174
St16	3	2234833	2120240	**St65**	3	2225969	2109429
St17	2	2243989	2074470	**St66**	3	2226616	2337326
St18	5	2092111	2224792	**St67**	2	2150986	2126469
St19	3	2079902	2070421	**St68**	3	2205830	2101553
St20	3	2151351	2189508	**St69**	1	2248029	2092343
St21	2	2261335	2205968	**St70**	3	2053470	2183114
St22	1	2198328	2096812	**St71**	1	2346086	2220162
St23	3	2151984	2196793	**St72**	2	2195573	2144439
St24	3	2066176	2341515	**St73**	2	2140360	2010591
St25	2	2581905	2441390	**St74**	2	2253357	2013947
St26	3	2238030	2341513	**St75**	2	2286985	2268781
St27	3	2133153	2198118	**St76**	2	2272345	2186992
St28	4	2029298	2193015	**St77**	4	2097762	2138903
St29	3	2213770	2067679	**St78**	4	2048797	2133184
St30	2	2124943	1979493	**St79**	1	2536930	2000218
St31	1	2174531	2548441	**St80**	3	2285851	2058445
St32	3	2190952	2948133	**St81**	3	2169217	2292690
St33	1	2681203	2446754	**St82**	1	2218347	2005371
St34	1	2301729	2170492	**St83**	3	2163200	2271453
St35	1	2156309	2181950	**St84**	3	2201554	2110309
St36	1	2155112	2001921	**St85**	4	2128020	2206571
St37	3	2072107	2717965	**St86**	2	2325872	2217457
St38	3	2060727	2128041	**St87**	2	2051594	2248561
St39	3	2199275	2251102	**St88**	1	2546491	2790062
St40	2	2046759	1998190	**St89**	2	2209026	2243091
St41	1	2108054	2027396	**St90**	2	2299968	2251586
St42	3	1953272	2056226	**St91**	2	2104646	2175519
St43	1	2068863	1786509	**St92**	3	2099608	2051958
St44	2	2357552	2103293	**St93**	3	1961799	2098980
St45	6	2134961	2246855	**St94**	3	2601138	2262041
St46	4	2118660	2369295	**St95**	3	2296307	2308837
St47	3	1876131	1768057	**St96**	5	2099865	2372730
St48	2	2774603	2258091	**St97**	4	2025357	2130097
St49	2	2084630	2066291	**St98**	2	2168715	2151094

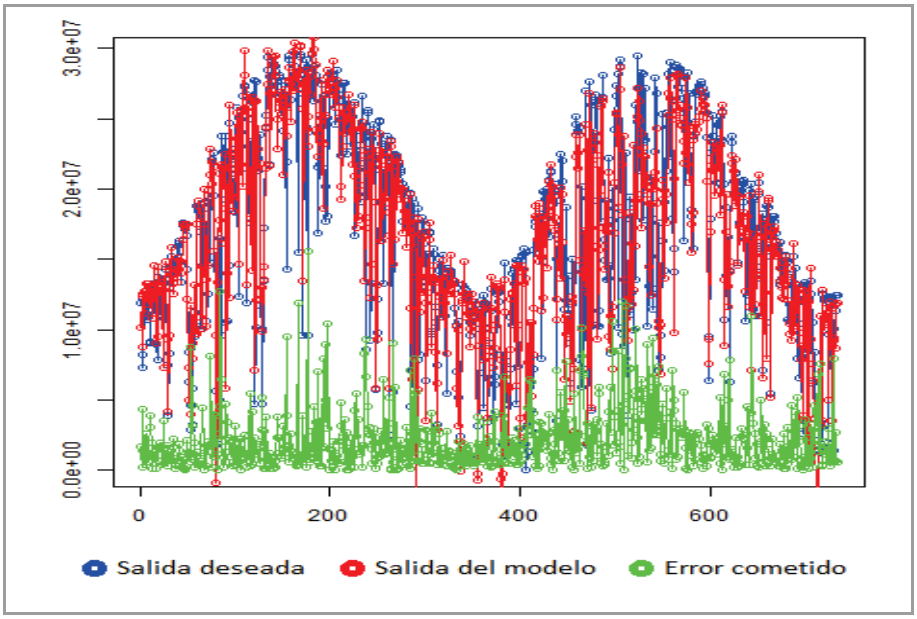
Tabla 3. Mejores resultados del experimento con LM.

Observando los resultados de la tabla vemos que LM obtiene los errores más bajos cuando utilizamos pocos puntos del GEFS (en la mayoría de las estaciones, el mejor error se obtiene utilizando 1, 2 o 3 puntos cercanos. El valor medio de puntos del GEFS utilizados es 2.4). Esto coincide con lo que dijimos en el apartado 2.1.1 sobre que si el modelo de regresión lineal cuenta con un número reducido de individuos, muestra una capacidad de generalización mayor que otros modelos más complejos.

Teniendo en cuenta todos los resultados de cada estación y no sólo los mejores, observamos en todos los casos que según aumentan los puntos cercanos a la estación utilizados, el error absoluto medio de entrenamiento tiende a bajar y el de test a subir. Por ello, podríamos decir que se produce sobreaprendizaje, ya que parece que los modelos "se aprenden" los datos de entrenamiento y pierden capacidad para generalizar. Podemos ver esta tendencia ilustrada en la Gráfica 2, que refleja las curvas de los errores de entrenamiento y test de la estación número 1.



Gráfica 2. Resultados de entrenamiento y test para la Estación 1 con LM.



Gráfica 3. Salida esperada frente a la salida generada por un modelo de LM.

4.2.2. Experimento 2: SVM Rbfdot

En este segundo experimento, pasamos a utilizar las Máquinas de Vectores de Soporte. Como ya explicamos, trabajaremos con diferentes kernel de las SVM. Cada kernel constituirá un experimento diferente, con su estudio de parámetros, pruebas y demás. El primero será el kernel Gaussiano o Rbfdot, como se denomina dicho kernel en R.

La función utilizada para crear los modelos de este experimento es *ksvm*, la cual ya ha sido explicada en el apartado 3.1.2 de este documento. El experimento es muy similar al anterior en cuanto a estructura. Los datos de entrada también se los pasamos a esta función en un *data.frame* y la descripción de la fórmula a ajustar es la misma (salida~.). La manera de crear los modelos es la misma que en el experimento 1 (el primer modelo con los 75 primeros atributos más la salida, el segundo con 150 más la salida, ...). La diferencia a nivel de código está evidentemente en la función para crear los modelos. En esta ocasión sí que indicaremos valores de los parámetros para los modelos. Como definimos en el capítulo de Diseño del Sistema, los parámetros que utilizaremos para este experimento son *C* (coste), *sigma* y *epsilon*. Lo primero que hay que hacer es ajustar el valor de dichos parámetros, y eso lo haremos con el estudio de parámetros que explicaré a continuación en el subapartado 4.2.2.1.

Una vez finalizado el ajuste de los parámetros, se realizarán las predicciones de entrenamiento y test y se calcularán sus respectivos MAE, siguiendo el mismo procedimiento que en el experimento anterior.

Este experimento también estará compuesto por 1568 modelos y 3136 predicciones en su totalidad.

4.2.2.1. Estudio de parámetros.

El estudio de parámetros para este experimento consistirá en una búsqueda exhaustiva en rejilla con los conjuntos de entrenamiento y validación creados (documentados en el apartado 4.1), para cada combinación de los parámetros *C*, *sigma* y *epsilon*.

Necesitamos elegir un rango de valores para cada parámetro. Dichos valores serán los que combinaremos para dar con los mejores valores para entrenar nuestros modelos. Los rangos que hemos elegido son:

C = (0.01, 0.03, 0.06, 0.12, 0.25, 0.50, 1.00, 2.00, 4.00).
sigma = (0.005, 0.010, 0.020, 0.040, 0.080, 0.160, 0.320, 0.640, 1.280).
epsilon = (0.01, 0.1, 0.5).

Iremos combinando todos estos valores y creando con ellos modelos, utilizando el conjunto de datos de entrenamiento de la primera estación con los 5 puntos del GEFS más cercanos a ella (creado en el punto 4.1), y con los modelos creados, realizaremos predicciones utilizando el conjunto de datos de validación para ver qué combinación de valores nos proporciona el MAE de validación más bajo. Dicha combinación de valores será la que utilizaremos en el experimento.

Una vez realizadas estas pruebas, la mejor combinación de parámetros resultante es *C=1*, *sigma=0.005* y *epsilon=0.1*, que da como resultado de validación un MAE de 2288496.343. Los valores de *C* y *epsilon* resultantes del ajuste son valores intermedios en sus respectivos rangos, pero el valor de *sigma* está en uno de los extremos del rango, siendo el valor más pequeño. Esto puede significar que si seguimos ajustando con valores más pequeños de *sigma*, podríamos obtener mejores resultados. Por ello, ampliaremos el estudio de parámetros añadiendo los valores *sigma = (0.001, 0.0005)*.

Tras combinar estos nuevos valores con los rangos del resto de parámetros, el MAE de validación más bajo obtenido es 2209064.164, y se consigue con los parámetros *C=1*, *sigma=0.001* y *epsilon=0.1*. Con la ampliación del estudio de parámetros hemos

podido mejorar los resultados, y ninguno de los valores ajustados está ya en los límites de los rangos, por lo que podemos dar por concluido el estudio de parámetros y quedarnos con estos valores.

4.2.2.2. Tiempos de ejecución.

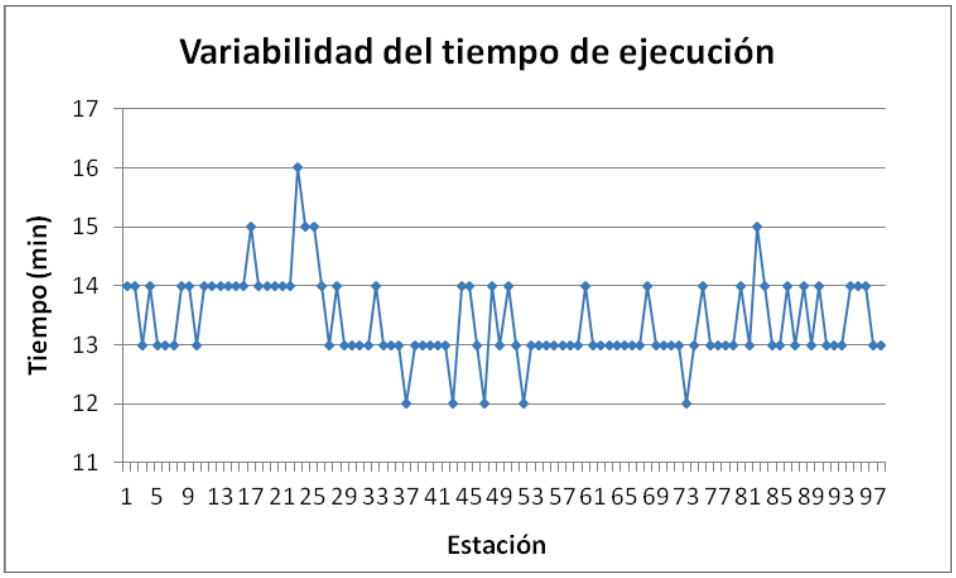
Los tiempos de ejecución de este experimento se han calculado igual que los del experimento anterior para poder compararlos posteriormente, por tanto, también incluyen todas las pruebas de cada estación.

En la Tabla 4, se muestran los tiempos de todas las estaciones, medidos también en minutos.

Estación	Tiempo	Estación	Tiempo	Estación	Tiempo	Estación	Tiempo
St01	14	**St26**	14	**St51**	13	**St76**	13
St02	14	**St27**	13	**St52**	12	**St77**	13
St03	13	**St28**	14	**St53**	13	**St78**	13
St04	14	**St29**	13	**St54**	13	**St79**	13
St05	13	**St30**	13	**St55**	13	**St80**	14
St06	13	**St31**	13	**St56**	13	**St81**	13
St07	13	**St32**	13	**St57**	13	**St82**	15
St08	14	**St33**	14	**St58**	13	**St83**	14
St09	14	**St34**	13	**St59**	13	**St84**	13
St10	13	**St35**	13	**St60**	14	**St85**	13
St11	14	**St36**	13	**St61**	13	**St86**	14
St12	14	**St37**	12	**St62**	13	**St87**	13
St13	14	**St38**	13	**St63**	13	**St88**	14
St14	14	**St39**	13	**St64**	13	**St89**	13
St15	14	**St40**	13	**St65**	13	**St90**	14
St16	14	**St41**	13	**St66**	13	**St91**	13
St17	15	**St42**	13	**St67**	13	**St92**	13
St18	14	**St43**	12	**St68**	14	**St93**	13
St19	14	**St44**	14	**St69**	13	**St94**	14
St20	14	**St45**	14	**St70**	13	**St95**	14
St21	14	**St46**	13	**St71**	13	**St96**	14
St22	14	**St47**	12	**St72**	13	**St97**	13
St23	16	**St48**	14	**St73**	12	**St98**	13
St24	15	**St49**	13	**St74**	13	**Media**	13,408
St25	15	**St50**	14	**St75**	14	**Desv. Típica**	0,701

Tabla 4. Tiempos de ejecución en minutos del experimento con SVM Rbfdot.

Podemos ver que los tiempos se han incrementado con respecto al primer experimento, pero siguen siendo tiempos que se pueden considerar bajos. Este incremento en los tiempos de ejecución se debe a que el algoritmo de predicción utilizado en este experimento es más complejo. El tiempo medio de ejecución por estación es aproximadamente 13 minutos y medio, y al igual que en el experimento anterior, tampoco hay variaciones considerables en los tiempos de ejecución entre estaciones. En la Gráfica 4 representamos la variabilidad del tiempo de ejecución entre estaciones de este método.



Gráfica 4. Variabilidad del tiempo de ejecución con SVM Rbfdot.

4.2.2.3. Problemas surgidos durante la experimentación.

A diferencia del experimento anterior, en éste sí surgieron ciertos problemas que hubo que solventar para poder concluir el experimento con éxito.

Las primeras pruebas realizadas presentaban resultados que no tenían ningún sentido, con errores de entrenamiento desorbitados. Decidimos probar otro paquete de R diferente a kernlab que también implementase SVM para comprobar si podía ser problema del paquete. Hicimos unas pequeñas pruebas con las SVM del paquete e1071[43], pero los resultados seguían siendo muy malos, similares a los de kernlab.

Finalmente, el problema resultó estar en los datos de entrada. Las SVM normalizan los atributos, y en los datos de entrada había columnas constantes, lo que provocaba que hubiese divisiones por

cero (máximo - mínimo = 0). Para solucionarlo, en el código programamos que no se tomasen las columnas constantes. Estas columnas no se podían eliminar del fichero de datos de entrenamiento directamente por varias razones, entre ellas porque sería mucho más lioso programar el número de columnas que se debían tomar para crear los modelos, ya que dejarían de ser 75 por punto del GEFS. Así que lo que hicimos fue añadir el código que eliminaba las columnas constantes después de haber tomado del fichero las columnas correspondientes al número de puntos utilizado para crear los modelos.

Con esto, los resultados de las predicciones estaban dentro de la normalidad, por lo que pudimos continuar con el experimento sin que surgieran más problemas significativos hasta su finalización.

4.2.2.4. Mejor resultado para cada estación.

El experimento concluye con los 3136 MAE que necesitamos. Al igual que en el experimento anterior, la Tabla 5 muestra los mejores resultados de cada estación con este método.

Estación	Puntos	MAE Train	MAE Test	Estación	Puntos	MAE Train	MAE Test
St01	7	1769861,05	1945451,18	**St50**	4	2035201,43	2098475,98
St02	6	1880913,97	2027958,97	**St51**	8	1818086,05	1826038,1
St03	5	1869735,26	2329632,5	**St52**	8	1563650,94	1896569,57
St04	11	1582917,66	1893320,5	**St53**	6	1888851,03	1872376,65
St05	6	1730418,25	1828343,23	**St54**	6	1865705,09	1973719,33
St06	3	1858958,39	2055035,62	**St55**	5	1893370,82	2025823,53
St07	8	1655409,6	1741668,86	**St56**	4	1966859,04	1860319,31
St08	5	1950498,53	1905479,15	**St57**	12	1567459,19	1898939,42
St09	6	1898532,91	1856038,33	**St58**	5	1918375,73	2007494,35
St10	4	1640788,43	1633619,12	**St59**	12	1553567,49	1780929,01
St11	5	1891759,13	1983910,7	**St60**	5	2160289,87	2307919,33
St12	4	1935371,92	1794996,68	**St61**	7	1751405,92	1927637,88
St13	7	1786045,56	1861366,41	**St62**	12	1670161,77	1898956,53
St14	10	1662309,94	1964387,44	**St63**	5	1958595,3	1963293,22
St15	4	1996528,87	1827009,77	**St64**	5	1867875,53	1829713,12
St16	1	2250898,92	1963724,98	**St65**	6	1909528,77	1896285,82
St17	7	1878470,11	1813784,51	**St66**	8	1838212,11	2105720,7
St18	5	1909261,95	2033148,6	**St67**	5	1864501,79	1950567,33
St19	4	1877333,53	1787204,29	**St68**	6	1926039,67	1832463,66
St20	6	1863795,81	2060015,42	**St69**	5	1919671,47	1808149,87
St21	6	1894507,07	1876254,45	**St70**	7	1744537,8	1994973,41
St22	6	1790754,01	1892479,27	**St71**	9	1846951,93	1893320,72

St23	9	1715976,61	1839749,45	**St72**	5	1888733,6	1853927,03
St24	12	1587873,6	2062277,13	**St73**	7	1743525,12	1763828
St25	6	2186147,32	2139730,28	**St74**	8	1836102,83	1862945,36
St26	6	1953071,88	2199890,7	**St75**	6	1935042,01	2065779,91
St27	6	1866779,83	1988928,54	**St76**	6	1909363,13	1936364,56
St28	5	1849041,48	1978471,42	**St77**	8	1764424,97	1953289,13
St29	5	1974366,2	1866284,4	**St78**	8	1723520,92	1907477,6
St30	7	1794166,32	1781637,02	**St79**	3	2197760,21	1745287,3
St31	3	1870761,99	2245816,67	**St80**	7	1974864,34	1823624,96
St32	8	1826077,64	2806908,9	**St81**	5	1948289,554	2102304,122
St33	3	2310625,64	2012156,83	**St82**	6	1834987,332	1747848,181
St34	4	2017793,93	1938993,34	**St83**	7	1822645,945	2077629,931
St35	6	1758674,01	1901288,39	**St84**	5	1949176,836	1956170,69
St36	12	1563578,6	1778792,87	**St85**	4	1988428,047	2126514,205
St37	8	1625006,11	2414802,04	**St86**	5	2036775,142	2045508,707
St38	7	1753489,23	1871350,21	**St87**	5	1801278,039	2020568,901
St39	6	1898775,89	2003174,91	**St88**	3	2258456,685	2714883,953
St40	9	1641855,07	1795833,95	**St89**	5	1921353,221	2004403,04
St41	6	1744474,1	1797761,96	**St90**	10	1821059,512	2048141,443
St42	9	1576705,19	1883335,49	**St91**	6	1800643,553	1943633,815
St43	4	1793873,62	1628461,06	**St92**	4	1907668,652	1850431,851
St44	4	2148338,86	2014603,62	**St93**	4	1823573,638	1871366,222
St45	6	1932260,14	1994821,67	**St94**	5	2302938,505	1988335,256
St46	7	1830536,12	2152285,81	**St95**	5	2044703,02	2141969,537
St47	3	1741753,24	1626090,95	**St96**	9	1776251,609	2140383,668
St48	5	2368551,86	1961618,12	**St97**	5	1804581,579	1868161,81
St49	6	1793679,66	1843529,18	**St98**	5	1878814,311	1884362,236

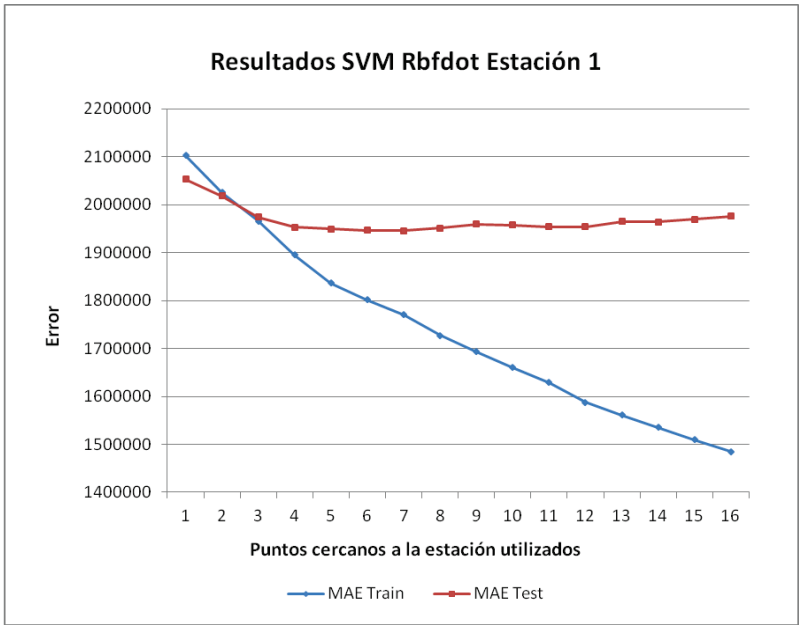
Tabla 5. Mejores resultados del experimento con SVM con kernel Gaussiano.

Los resultados de este experimento mejoran los del anterior. Con éste método de predicción, los modelos necesitan más tiempo de entrenamiento, pero proporcionan una mayor precisión.

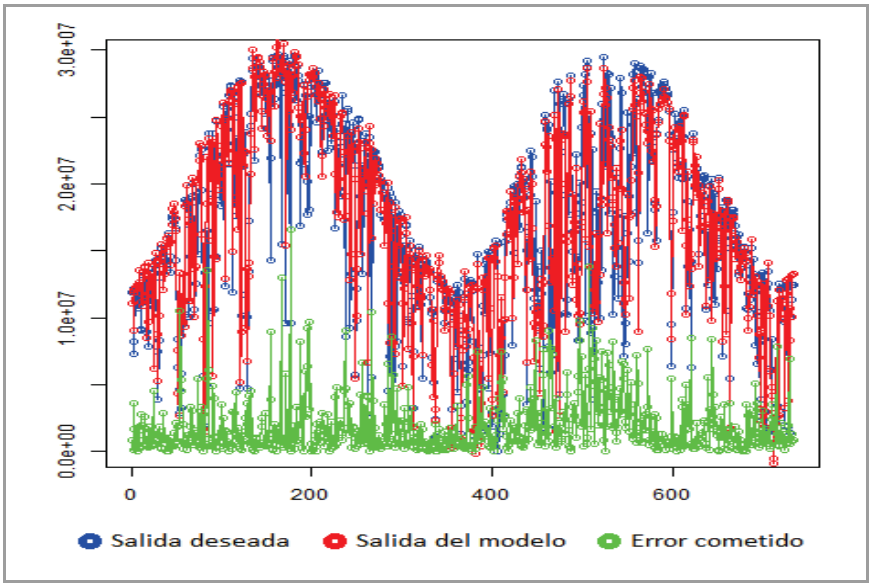
El número de puntos cercanos a la estación necesarios para obtener los mejores resultados aumenta con respecto al experimento anterior. La mayoría de las estaciones obtienen el error más bajo utilizando entre 4 y 8 puntos cercanos (el valor medio de puntos del GEFS utilizados para conseguir el MAE más bajo es 6.16).

Analizando todos los datos por estación en lugar de sólo los mejores, tal y como vemos en la Gráfica 5 que se muestra a continuación, el error absoluto medio de entrenamiento muestra una clara tendencia a mejorar cuantos más atributos de entrada sean utilizados. Esto es totalmente lógico, pues el modelo puede aprender cada vez mejor los datos de entrada. Por su parte,

el error absoluto medio de test tiende a mejorar progresivamente hasta el caso en el que se utilizan los 7 puntos del GEFS más cercanos a la estación (la estación 1 en este caso). De ahí en adelante, el error comienza a empeorar. El aumento del error no es continuo, algún caso mejora el error del anterior (como el de 11 puntos que mejora al de 10), pero la tendencia del error es claramente a empeorar.



Gráfica 5. Resultados de entrenamiento y test para la Estación 1 con SVM Rbfdot.



Gráfica 6. Salida esperada frente a la salida generada por un modelo de SVM Rbfdot.

4.2.3. Experimento 3: SVM Polydot

El este tercer experimento, seguiremos utilizando las Máquinas de Vectores de Soporte, pero en esta ocasión con kernel polinómico en lugar de Gaussiano.

Estructuralmente, todo el experimento es idéntico al anterior. Lo único que cambia es la creación de los modelos, ya que al cambiar el kernel utilizado, los parámetros son otros. En concreto, los parámetros utilizados en este experimento son *C* (Coste), *epsilon* y *degree* (grado del polinomio). Todo lo demás, incluidas las predicciones, es idéntico al experimento anterior.

4.2.3.1. Estudio de parámetros.

Igual que en el estudio de parámetros anterior, realizaremos una búsqueda exhaustiva en rejilla para ajustar los valores de los parámetros de la función.

El parámetro *degree* decidimos utilizarlo sólo con valor 2 por las razones argumentadas en el apartado 3.1.2, así que dicho parámetro no será incluido en el estudio de parámetros. Por tanto, los parámetros que debemos ajustar en este estudio son *C* y *epsilon*. Ambos parámetros estaban incluidos en el experimento anterior, y al tratarse de SVM también, los rangos de valores utilizados serán los mismos:

C = (0.01, 0.03, 0.06, 0.12, 0.25, 0.50, 1.00, 2.00, 4.00). epsilon = (0.01, 0.1, 0.5).

Tras realizar las pruebas con todas las combinaciones posibles de estos dos parámetros, el mejor resultado se obtiene con *C=0.01* y *epsilon=0.5*, que dan como resultado un MAE de validación de 3392037.173. Los valores resultantes para ambos parámetros se sitúan en alguno de los extremos de los rangos. El valor de *C* es el más bajo de su rango y el de *epsilon* es el más alto del suyo. Esto nos indica que debemos ampliar los rangos y seguir ajustando los parámetros, porque seguramente podamos conseguir mejorar el error.

Lo primero que probamos fueron valores de *epsilon* más grandes que 0.5 con el valor de *C=0.01* para observar el comportamiento del mejor valor de *C* si incrementaba *epsilon*. Los nuevos valores de *epsilon* que probamos fueron *epsilon = (0.75, 1)*. Resultó que con *epsilon=0.75* el error mejoraba hasta 3265655.788, por lo que sí merecía la pena ampliar el rango de valores de *epsilon* hasta el final

del ajuste. Lo siguiente fue ampliar el rango de *C* con valores inferiores a 0.01. Los valores que añadimos al rango fueron *C = (0.001, 0.005)*.

Tras combinar estos nuevos valores con el rango de *epsilon*, el MAE de validación bajó hasta 2880302.435, una mejora bastante considerable, y se conseguía con *epsilon=0.5* y *C=0.001*, el nuevo extremo inferior del rango de valores de *C*, por lo que debíamos seguir ampliando dicho rango con valor inferiores. Esta vez añadimos los valores *C = (0.0001, 0.0005)*. Con estos nuevos valores, el MAE de validación siguió bajando hasta 2496249.351, y nuevamente se obtenía con el valor más pequeño de *C*, por lo que seguimos probando con valores más bajos: *C = (0.00001, 0.00005)*.

Finalmente, con estos últimos valores añadidos al rango de *C* pudimos poner fin a este ajuste de parámetros. El mejor MAE de validación obtenido fue 2414566.241, y se consiguió con *C=0.00005* y *epsilon=0.1*. Ninguno de estos valores era ya un extremo de rango, por lo que dimos por finalizado el estudio de parámetros para las SVM con kernel polinómico, eligiendo estos valores de *C* y *epsilon* para realizar las pruebas finales del experimento.

4.2.3.2. Tiempos de ejecución.

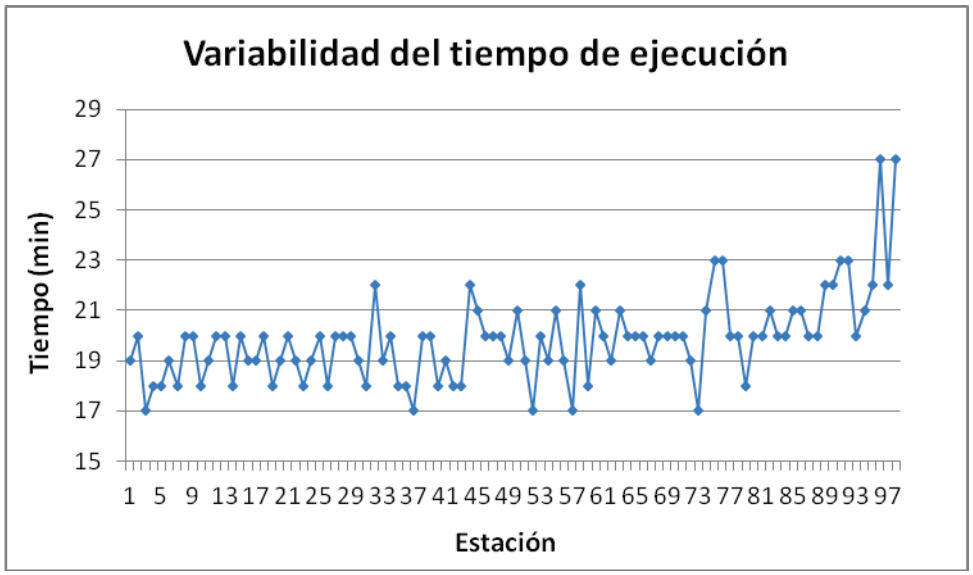
Como en los casos anteriores, se recogieron los tiempos que necesitaba R para ejecutar todas las pruebas de cada estación. Dichos tiempos se muestran en la Tabla 6, medidos en minutos.

Estación	Tiempo	Estación	Tiempo	Estación	Tiempo	Estación	Tiempo
St01	19	**St26**	18	**St51**	19	**St76**	23
St02	20	**St27**	20	**St52**	17	**St77**	20
St03	17	**St28**	20	**St53**	20	**St78**	20
St04	18	**St29**	20	**St54**	19	**St79**	18
St05	18	**St30**	19	**St55**	21	**St80**	20
St06	19	**St31**	18	**St56**	19	**St81**	20
St07	18	**St32**	22	**St57**	17	**St82**	21
St08	20	**St33**	19	**St58**	22	**St83**	20
St09	20	**St34**	20	**St59**	18	**St84**	20
St10	18	**St35**	18	**St60**	21	**St85**	21
St11	19	**St36**	18	**St61**	20	**St86**	21
St12	20	**St37**	17	**St62**	19	**St87**	20
St13	20	**St38**	20	**St63**	21	**St88**	20
St14	18	**St39**	20	**St64**	20	**St89**	22
St15	20	**St40**	18	**St65**	20	**St90**	22
St16	19	**St41**	19	**St66**	20	**St91**	23
St17	19	**St42**	18	**St67**	19	**St92**	23

St18	20	**St43**	18	**St68**	20	**St93**	20
St19	18	**St44**	22	**St69**	20	**St94**	21
St20	19	**St45**	21	**St70**	20	**St95**	22
St21	20	**St46**	20	**St71**	20	**St96**	27
St22	19	**St47**	20	**St72**	19	**St97**	22
St23	18	**St48**	20	**St73**	17	**St98**	27
St24	19	**St49**	19	**St74**	21	**Media**	19,857
St25	20	**St50**	21	**St75**	23	**Desv. Típica**	1,77

Tabla 6. Tiempos de ejecución en minutos del experimento con SVM Polydot.

De nuevo observamos un incremento en los tiempos de ejecución con respecto a los experimentos anteriores. En comparación con el experimento 2 (SVM Rbfdot), el incremento se puede estimar en alrededor de un 40% más de tiempo al pasar de kernel Gaussiano a polinómico. El tiempo medio de ejecución por estación es aproximadamente 20 minutos. En cuanto a la variabilidad del tiempo de ejecución entre estaciones (representada en la Gráfica 7), los tiempos de todas las estaciones oscilan entre 17 y 23 minutos, salvo las estaciones 96 y 98, que tardaron 27 minutos en ejecutarse las pruebas de cada una. Esto no parece que se deba a características de los datos de dichas estaciones, sino a que en ese momento el equipo estaría empleando menos recursos para la ejecución de las pruebas porque estaría realizando otras tareas simultáneamente, por lo que podríamos considerar estos pequeños picos como ruido.



Gráfica 7. Variabilidad del tiempo de ejecución con SVM Polydot.

4.2.3.3. Problemas surgidos durante la experimentación.

En este experimento, al seguir utilizando Máquinas de Vectores de Soporte, el problema con las columnas constantes sigue presente. Esta vez no supuso un problema porque ya quedó resuelto en el experimento anterior y la solución, evidentemente, se aplica en este caso también, pues se debía a un conflicto de las SVM con los datos de entrada, nada que ver con el kernel que es lo único que cambia en este experimento.

Por lo demás, esta experimentación pudo llevarse a cabo sin ningún tipo de problema significativo. Los tiempos de ejecución se incrementaron con respecto a los experimentos anteriores, pero no eran tiempos tan grandes como para suponer un problema en cuanto a recursos computacionales.

4.2.3.4. Mejor resultado para cada estación.

Tras finalizar la experimentación, obtuvimos los 3136 valores de MAE de entrenamiento y test que buscábamos. En la siguiente tabla, como en los casos anteriores, mostramos los mejores resultados obtenidos para cada una de las estaciones.

Estación	Puntos	MAE Train	MAE Test	Estación	Puntos	MAE Train	MAE Test
St01	2	1994669,48	2157422,98	**St50**	3	1960246,985	2290077,015
St02	2	2085083,75	2252130,59	**St51**	2	2085813,97	2021141,214
St03	3	1827826,54	2571622,94	**St52**	3	1655058,799	2083583,779
St04	2	1947465,88	2085418,11	**St53**	3	1923906,228	2106491,9
St05	2	1930846,88	2046042,19	**St54**	3	1897118,768	2135744,582
St06	2	1874934,62	2191779,99	**St55**	3	1889568,31	2235266,89
St07	2	1880617,67	1892295,16	**St56**	3	1863784,531	1969156,004
St08	4	1781870,35	2096679,45	**St57**	2	1947329,091	2105397,691
St09	2	2100463,57	2000388,87	**St58**	3	1911918,903	2107190,086
St10	2	1687152,44	1809694,32	**St59**	3	1773944,569	1987868,524
St11	3	1864816,49	2184308,46	**St60**	3	2124429,847	2439665,716
St12	3	1851746,32	1974331,34	**St61**	2	2017292,12	2077034,237
St13	3	1871742,75	2082499,65	**St62**	2	2101003,307	2134091,012
St14	2	1998720,06	2073601,63	**St63**	2	2111888,348	2157376,219
St15	2	2106267,033	1988756,38	**St64**	3	1845864,85	2040476,673
St16	2	2121634,211	2135624,892	**St65**	2	2084416,46	2081893,853
St17	2	2104769,052	1951399,838	**St66**	3	1926644,168	2262443,771
St18	2	2063077,653	2243061,947	**St67**	2	2003894,131	2153332,734
St19	4	1672557,075	1924886,493	**St68**	3	1950044,55	2060020,88

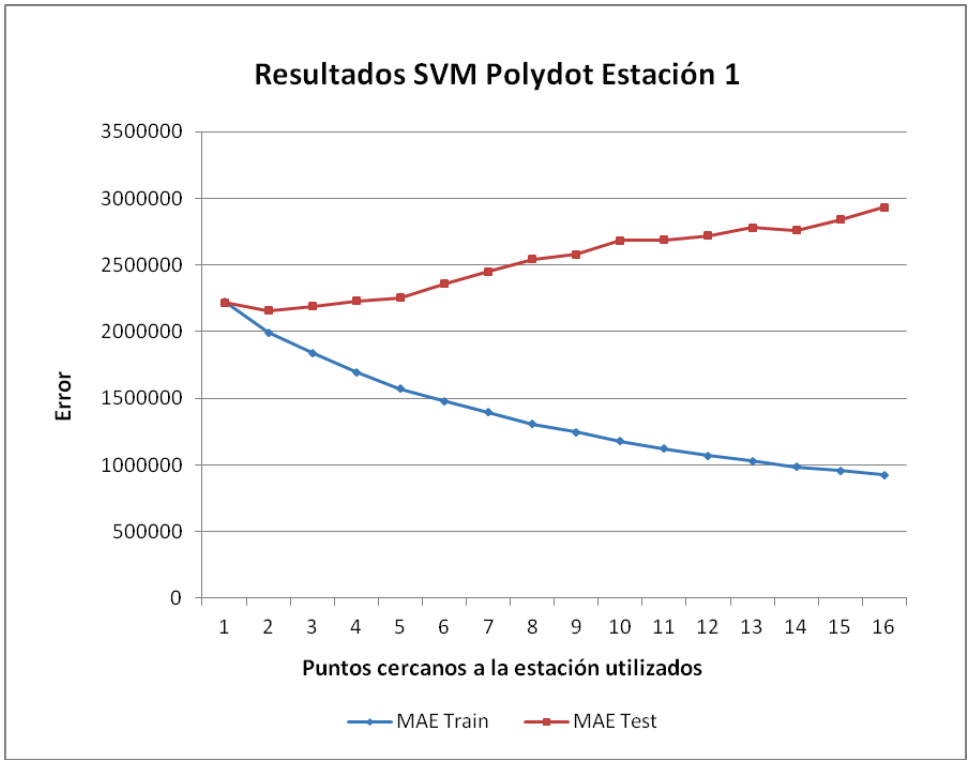
St20	3	1894937,711	2253920,975	**St69**	2	2093219,794	2047154,142
St21	3	1922516,859	2109173,292	**St70**	3	1804184,482	2126995,983
St22	3	1804090,294	2069521,646	**St71**	3	1977153,873	2074870,55
St23	2	2031121,74	1983079,803	**St72**	2	2040852,652	2074347,037
St24	3	1818480,081	2256434,857	**St73**	3	1766733,332	1922191,2
St25	4	2088225,265	2365356,774	**St74**	3	1908343,126	1999965,57
St26	3	1967824,634	2294001,738	**St75**	2	2106670,488	2158136,647
St27	2	2067828,073	2072943,526	**St76**	2	2062519,332	2042738,534
St28	4	1698098,346	2118905,167	**St77**	3	1876467,518	2115469,891
St29	3	1942721,809	2093142,744	**St78**	2	2017308,63	2073797,224
St30	2	2006459,061	1966718,339	**St79**	3	2053599,783	1885724,337
St31	2	1911629,882	2404796,758	**St80**	3	2016654,391	2037416,291
St32	3	1914052,844	2912824,034	**St81**	3	1919989,166	2217275,686
St33	2	2361172,799	2184587,6	**St82**	2	2047825,732	1964912,67
St34	2	2111358,072	2077762,513	**St83**	2	2072377,126	2310906,152
St35	3	1770550,032	2100460,033	**St84**	2	2080499,272	2159100,858
St36	2	1947619,877	1954532,425	**St85**	2	2116124,637	2219865,704
St37	4	1626228,979	2521336,23	**St86**	2	2171905,466	2140406,16
St38	3	1821305,129	2078765,211	**St87**	3	1748825,864	2243541,296
St39	3	1927648,085	2165432,302	**St88**	2	2302113,487	2974078,935
St40	4	1651836,359	1950736,775	**St89**	3	1907936,533	2143074,813
St41	2	1930151,064	1994224,616	**St90**	2	2141483,286	2281789,368
St42	2	1845146,903	2114209,937	**St91**	2	1996617,773	2081644,818
St43	2	1841591,583	1833762,552	**St92**	3	1849508,12	2041106,331
St44	2	2220898,447	2112658,558	**St93**	4	1643809,633	2030795,356
St45	4	1830311,439	2108124,981	**St94**	4	2150489,274	2134143,446
St46	2	2061302,128	2239479,181	**St95**	3	2029617,057	2281444,729
St47	2	1757933,928	1817564,941	**St96**	4	1775642,825	2262034,53
St48	2	2506958,876	2180457,544	**St97**	2	1942098,254	2035741,893
St49	2	1957193,739	1965726,723	**St98**	3	1876448,977	2053361,392

Tabla 7. Mejores resultados del experimento con SVM con kernel polinómico.

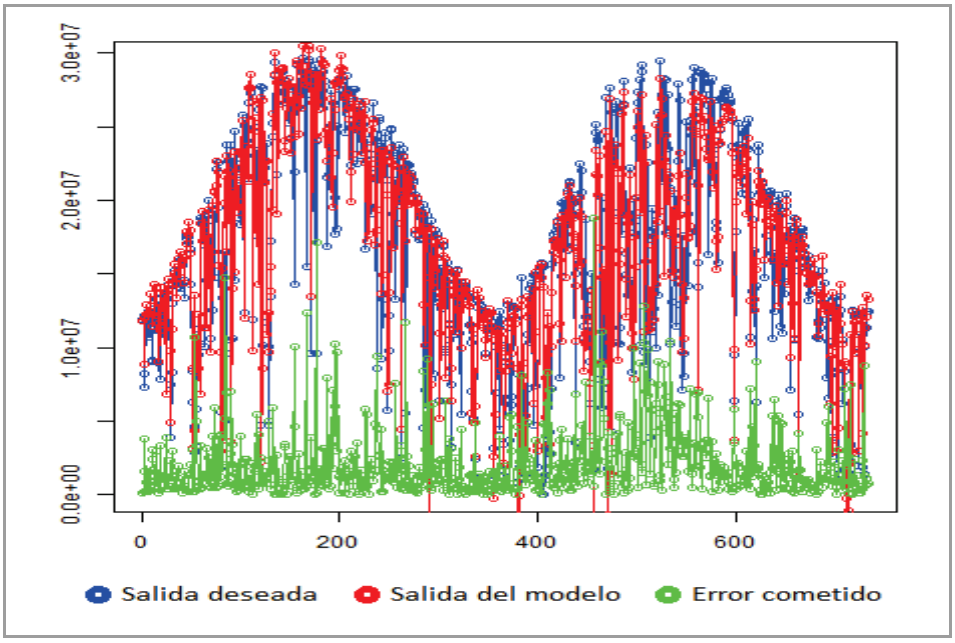
En la tabla podemos ver que con el cambio de kernel de Gaussiano a polinómico hemos empeorado los resultados. También hay una diferencia con respecto al kernel Gaussiano en cuanto a la cantidad de datos de entrada. Con todas la estaciones, se obtienen los mejores resultados utilizando entre 2 y 4 puntos del GEFS (el valor medio de puntos del GEFS utilizados para obtener el menor MAE es 2.6).

Observando todos los resultados de las estaciones, la evolución del error es bastante parecida a la del experimento con LM. Fijándonos

en la Gráfica 8, que representa los resultados de todos los modelos entrenados para la estación 1, podemos ver que el error absoluto medio de entrenamiento disminuye al aumentar el número de puntos cercanos utilizados, mientras que el MAE de test muestra una ligera mejora al aumentar los puntos utilizados de 1 a 2, pero después tiende a empeorar y el error aumenta. Por tanto, podemos decir que se produce sobreaprendizaje a partir de ese punto, pues el error de entrenamiento cae mientras el de test no deja de aumentar.



Gráfica 8. Resultados de entrenamiento y test para la Estación 1 con SVM Polydot.



Gráfica 9. Salida esperada frente a la salida generada por un modelo de SVM Polydot.

4.2.4. Experimento 4: SVM Tanhdot

Este experimento será el último en el que utilicemos las Máquinas de Soporte Vectorial. El kernel empleado esta vez será el sigmoidal. Aquí también mantendremos la estructura de los experimentos con SVM, y de nuevo lo único que varía es el kernel utiizado.

4.2.4.1. Estudio de parámetros.

Los parámetros utilizados en este caso vuelven a ser *C* y *epsilon*. El resto toman su valor por omisión, pues lo que queremos ver es cómo se comportan los datos con este kernel. Los rangos vuelven a ser los mismos:

C = (0.01, 0.03, 0.06, 0.12, 0.25, 0.50, 1.00, 2.00, 4.00). epsilon = (0.01, 0.1, 0.5).

Tras hacer las combinaciones de valores con la búsqueda exhaustiva en rejilla que estamos utilizando en todos los experimentos para ajustar los parámetros, los resultados que obtenemos son muy pobres con respecto a los que hemos venido obteniendo en los experimentos anteriores. Por ello, decidimos que no merecía la pena seguir con el estudio, porque por mucho que consiguiéramos mejorar los resultados, no llegarían a ser interesantes para nuestro estudio.

4.2.4.2. Problemas surgidos durante la experimentación.

Tras los resultados obtenidos durante el ajuste de parámetros, decidimos no seguir adelante con este experimento, pues no nos iba a aportar datos relevantes para nuestro estudio. De este modo, el experimento 4 queda finalizado en este punto.

4.2.5. Experimento 5: GBM

En este nuevo experimento, dejamos de lado las SVM para pasar a utilizar las técnicas de *Boosting*. Como ya se explicó en el capítulo 3 de esta memoria, utilizaremos dos paquetes de R diferentes que implementan este tipo de técnicas. El primero de ellos es el paquete GBM, explicado en el punto 3.1.3.

Para crear los modelos, utilizaremos la función *gbm*, perteneciente al paquete homónimo. Para más detalles sobre la función, se puede volver al apartado 3.1.3, donde se explicaba la parte técnica con todo detalle. La creación de los modelos mantiene la estructura habitual de nuestros experimentos. A la función *gbm* le pasamos la descripción de la fórmula a ajustar (*salida~.*) y los datos de entrada en un *data.frame*. Dichos datos varían según el modelo que toque crear, tal y como ya hemos explicado en los experimentos anteriores (para crear el modelo con el punto más cercano, se cogen las 75 primeras columnas y la salida; y se van incrementando las columnas de 75 en 75 al aumentar los puntos de GEFS cercanos a la estación que queremos utilizar, hasta llegar al decimosexto y último modelo, en el que utilizamos los 16 puntos y tomamos el fichero completo).

Además de eso, debemos pasar a la función valores para los parámetros *n.trees, shrinkage, interaction.depth* y *distribution*, como ya quedó estipulado en el capítulo del diseño del sistema.

Una vez hayamos creado los modelos, realizaremos las predicciones. Esto se hará con la función *predict*, de forma similar a los experimentos anteriores, solo que en este caso además de pasarle el modelo y los datos de test, tendremos que pasarle el número de árboles, como ya explicamos en el punto 3.1.3. Después, calcularemos el error absoluto medio de cada predicción. Como en los anteriores experimentos, este tendrá un total de 1568 modelos y 3136 predicciones.

4.2.5.1. Estudio de parámetros.

El artículo[12] del que parte este Trabajo de Fin de Grado también empleaba esta técnica. Gracias al artículo, sabemos de antemano que esta técnica tarda bastante en entrenar los modelos. Esto se debe a

que utiliza árboles de decisión, lo que ralentiza el proceso y el modelo aprende más lento pero más preciso.

Debido a esto, hemos decidido no realizar ajuste de parámetros para este experimento y utilizar los valores que se emplearon en el estudio del artículo, que sabemos que son óptimos porque en él sí se realizó estudio de parámetros para este caso. Los valores de los parámetros son *n.trees=5000*, *shrinkage=0.01*, *interaction.depth=10* y *distribution="laplace"*.

4.2.5.2. Tiempos de ejecución.

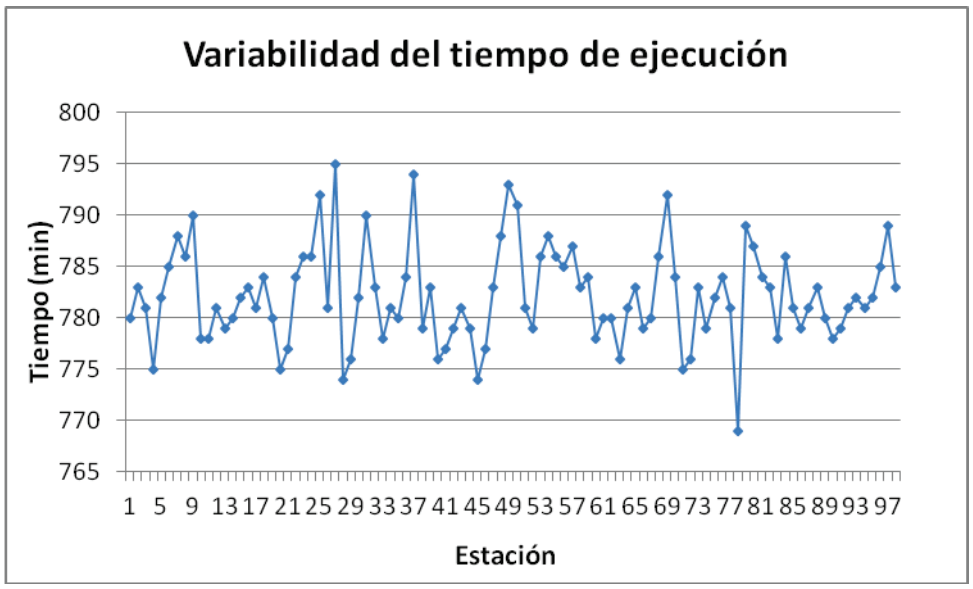
Como es habitual en nuestros experimentos, hemos calculado los tiempos de ejecución de las pruebas de cada una de las 98 estaciones que estamos estudiando con esta técnica. En la Tabla 8, los mostramos calculados en minutos.

Estación	Tiempo	Estación	Tiempo	Estación	Tiempo	Estación	Tiempo
St01	780	**St26**	781	**St51**	781	**St76**	784
St02	783	**St27**	795	**St52**	779	**St77**	781
St03	781	**St28**	774	**St53**	786	**St78**	769
St04	775	**St29**	776	**St54**	788	**St79**	789
St05	782	**St30**	782	**St55**	786	**St80**	787
St06	785	**St31**	790	**St56**	785	**St81**	784
St07	788	**St32**	783	**St57**	787	**St82**	783
St08	786	**St33**	778	**St58**	783	**St83**	778
St09	790	**St34**	781	**St59**	784	**St84**	786
St10	778	**St35**	780	**St60**	778	**St85**	781
St11	778	**St36**	784	**St61**	780	**St86**	779
St12	781	**St37**	794	**St62**	780	**St87**	781
St13	779	**St38**	779	**St63**	776	**St88**	783
St14	780	**St39**	783	**St64**	781	**St89**	780
St15	782	**St40**	776	**St65**	783	**St90**	778
St16	783	**St41**	777	**St66**	779	**St91**	779
St17	781	**St42**	779	**St67**	780	**St92**	781
St18	784	**St43**	781	**St68**	786	**St93**	782
St19	780	**St44**	779	**St69**	792	**St94**	781
St20	775	**St45**	774	**St70**	784	**St95**	782
St21	777	**St46**	777	**St71**	775	**St96**	785
St22	784	**St47**	783	**St72**	776	**St97**	789
St23	786	**St48**	788	**St73**	783	**St98**	783
St24	786	**St49**	793	**St74**	779	**Media**	782,163
St25	792	**St50**	791	**St75**	782	**Desv. Típica**	4,718

Tabla 8. Tiempos de ejecución en minutos del experimento con GBM.

Se repite el patrón del incremento del tiempo de ejecución con respecto a los experimentos anteriores, pero en esta ocasión el incremento es mucho mayor, pasando de varios minutos a bastantes horas. El tiempo medio de ejecución por estación es de aproximadamente 782 minutos, lo que equivale a alrededor de 13 horas. Cada modelo tarda casi una hora en ser entrenado, incrementándose el tiempo necesario al aumentar la cantidad de datos de entrada. Este aumento del tiempo de ejecución se debe al algoritmo utilizado. GBM utiliza árboles de decisión para ajustar los modelos, lo que requiere mucho más tiempo de entrenamiento que el resto de técnicas utilizadas en el estudio.

En la Gráfica 10 está representada la variabilidad del tiempo de ejecución por estación de GBM. En ella no vemos nada inusual, pues no hay grandes variaciones entre estaciones teniendo en cuenta las cifras entre las que se mueven los tiempos (entre 770 y 795 minutos).



Gráfica 10. Variabilidad del tiempo de ejecución con GBM.

4.2.5.3. Problemas surgidos durante la experimentación.

El principal problema que surgió durante esta experimentación fue la cantidad de recursos que demandaba.

Cada modelo tardaba casi una hora en entrenarse, y según se iba incrementando la cantidad de atributos utilizados para el entrenamiento, tardaba aún más. Se requería de media unas 13 horas para completar el estudio de una estación. Esto suponía una estimación de 1274 horas de cómputo para completar el estudio de

las 98 estaciones. Utilizando un único equipo para lanzar las pruebas, supondría aproximadamente 53.1 días de ejecución de pruebas ininterrumpidas, y esto sin tener en cuenta problemas imprevistos. Además de requerir mucho tiempo de cómputo, también requería más volumen de memoria que los experimentos anteriores debido a la utilización de los árboles de decisión.

La alternativa de simplificar el modelo no era una opción, puesto que de esa manera los resultados obtenidos serían peores, y nuestro objetivo es estudiar las mejores técnicas.

La opción que llevamos a cabo fue utilizar más de un equipo para ejecutar las pruebas de manera simultánea. Utilicé un total de 3 equipos para lanzar las pruebas simultáneamente. Aun así, esto ralentizó considerablemente el proyecto, pero al menos se pudo finalizar este experimento con éxito.

4.2.5.4. Mejor resultado para cada estación.

Tras encontrar la solución para poder finalizar las pruebas de este experimento, pudimos obtener las 3136 predicciones que necesitábamos. En la Tabla 9, se recogen los mejores MAE de cada estación calculados a partir de esas predicciones.

Estación	Puntos	MAE Train	MAE Test	Estación	Puntos	MAE Train	MAE Test
St01	6	1218353	1855707	**St50**	12	1247982	2028098
St02	10	1233574	1975037	**St51**	8	1273543	1757787
St03	13	1240658	2233396	**St52**	9	1083441	1820596
St04	13	1143164	1811303	**St53**	5	1311254	1814088
St05	15	1083062	1752662	**St54**	13	1250648	1861849
St06	11	1111849	1965335	**St55**	4	1276287	1995121
St07	13	1139474	1677878	**St56**	14	1183759	1811945
St08	4	1329200	1925517	**St57**	14	1174100	1819292
St09	12	1273340	1842648	**St58**	16	1228098	1984560
St10	15	987700,7	1578026	**St59**	13	1143017	1743415
St11	8	1259835	1908876	**St60**	10	1318328	2283229
St12	15	1153999	1790856	**St61**	15	1165599	1912411
St13	10	1233575	1852685	**St62**	5	1335239	1839134
St14	14	1208450	1843218	**St63**	7	1306001	1976304
St15	6	1314642	1859312	**St64**	7	1230733	1804579
St16	12	1276361	1896294	**St65**	15	1234839	1892455
St17	15	1255916	1729004	**St66**	13	1269406	2121665
St18	11	1215610	1971418	**St67**	13	1170603	1829510

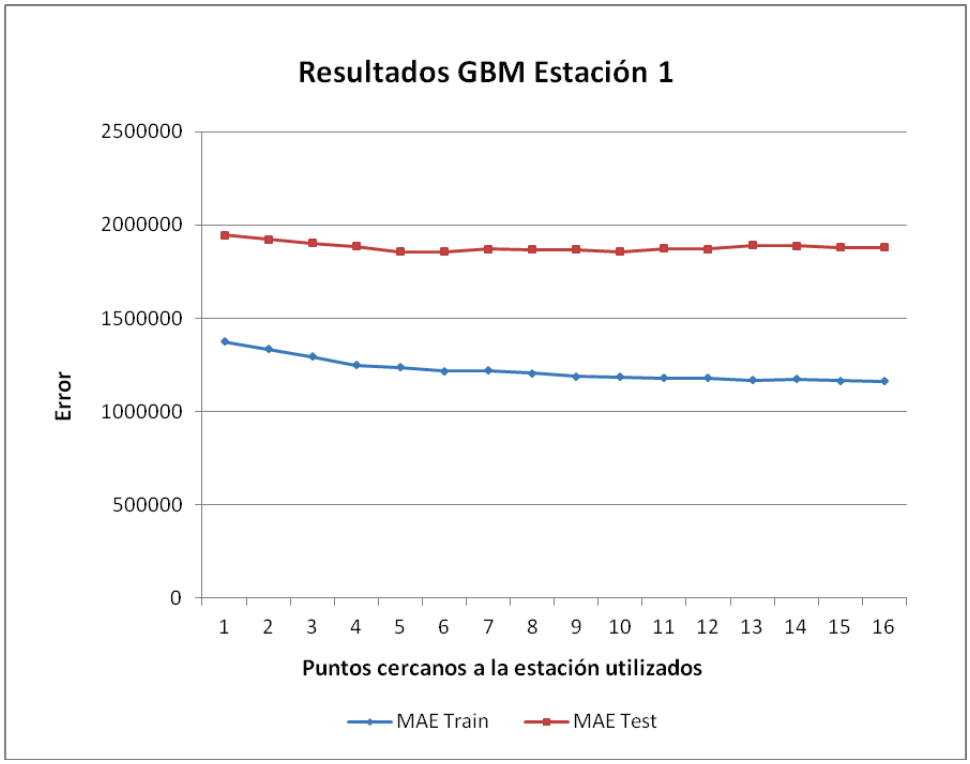
St19	12	1161421	1709213	**St68**	7	1291884	1836829
St20	14	1220331	2008656	**St69**	6	1282741	1772268
St21	16	1208997	1896187	**St70**	14	1159070	2019404
St22	10	1157120	1860242	**St71**	15	1262168	1816801
St23	13	1196537	1768819	**St72**	16	1195168	1825377
St24	4	1268081	1976340	**St73**	12	1192689	1632463
St25	6	1440800	2178541	**St74**	16	1214039	1801162
St26	9	1273358	2144418	**St75**	16	1250400	2008577
St27	13	1192618	1956417	**St76**	10	1278502	1784806
St28	16	1180569	1946896	**St77**	15	1196469	1934995
St29	11	1252757	1834726	**St78**	14	1183164	1886889
St30	15	1181275	1729986	**St79**	13	1375286	1704511
St31	15	1139497	2177457	**St80**	14	1335315	1765568
St32	8	1252564	2749588	**St81**	15	1227288	2069005
St33	11	1398759	1957243	**St82**	5	1274633	1711124
St34	15	1257111	1971086	**St83**	7	1259886	2051509
St35	11	1151281	1803274	**St84**	16	1213934	1910163
St36	13	1160637	1717042	**St85**	9	1232104	2121794
St37	8	1188214	2302693	**St86**	3	1382052	2035380
St38	7	1210104	1818631	**St87**	3	1243115	1915459
St39	5	1293642	2005919	**St88**	8	1428645	2647134
St40	15	1147142	1691198	**St89**	12	1235173	2023380
St41	12	1154142	1684255	**St90**	12	1303322	1962978
St42	12	1088834	1854158	**St91**	15	1163910	1841384
St43	16	1061840	1576497	**St92**	9	1214732	1777864
St44	9	1331178	1972399	**St93**	10	1130797	1758558
St45	6	1302693	2008921	**St94**	11	1480792	1979587
St46	6	1258106	2157862	**St95**	9	1310266	2129028
St47	16	1024205	1605469	**St96**	7	1246030	2109794
St48	16	1501242	1900945	**St97**	13	1134634	1806760
St49	15	1157462	1776019	**St98**	10	1227473	1890032

Tabla 9. Mejores resultados del experimento con GBM.

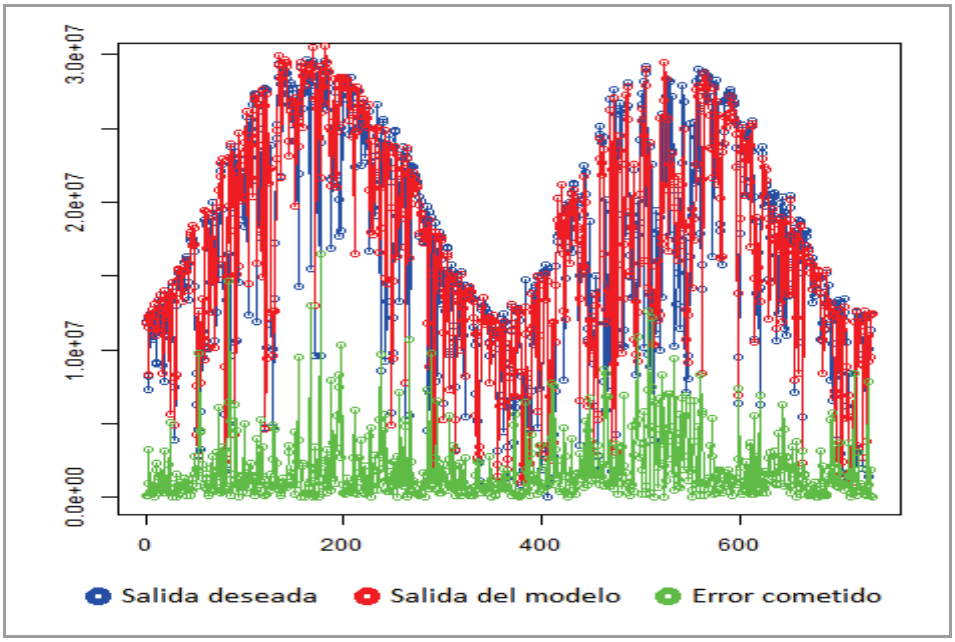
Los resultados obtenidos en este experimento son los mejores conseguidos hasta ahora. El mejor MAE de test conseguido en todas las estaciones es el más bajo de todos los métodos. La combinación de algoritmos en el proceso de entrenamiento de modelos de GBM consigue predicciones más precisas, aunque aprende de manera mucho más lenta.

Por otro lado, el número de puntos cercanos con el que cada estación consigue los mejores resultados es bastante dispar. Hay muy pocas estaciones que consiguen el mejor resultado con un número bajo de puntos. En la mayoría de estaciones se consiguen las mejores predicciones utilizando entre 12 y 16 puntos cercanos, casi la totalidad de los datos disponibles (el valor medio de puntos del GEFS utilizados para obtener los mejores resultados es 11.15).

En cuanto a la variabilidad del error, al tener en cuenta todos los datos de la estación en lugar de sólo el mejor, tal y como podemos ver en la Gráfica 11 que representa la estación 1, las curvas de MAE de entrenamiento y test son muy similares. El error de entrenamiento se mueve en un rango de valores menor que el de test, pero el comportamiento de ambos errores al aumentar el número del puntos del GEFS utilizados en el entrenamiento es el mismo. El error más alto es el obtenido al principio, con la menor cantidad de datos de entrada utilizados, y ambos errores tienden a mejorar cuando el volumen de los datos de entrada aumenta. La mejora no es continua, pues no se consigue el mejor resultado con los 16 puntos en todas las estaciones, pero los resultados de todos los modelos son muy similares, de modo que la tendencia parece indicar en la mayoría de los casos que los resultados podrían seguir mejorando si utilizásemos más de 16 puntos del GEFS para entrenar los modelos. Aunque como ya he dicho, los resultados de todos los modelos son muy similares, y dada la cantidad de recursos y tiempo que necesita GBM para entrenar los modelos cuando aumentan los datos de entrada, seguramente no merecería la pena utilizar más puntos del GEFS para realizar las predicciones.



Gráfica 11. Resultados de entrenamiento y test para la Estación 1 con GBM.



Gráfica 12. Salida esperada frente a la salida generada por un modelo de GBM.

4.2.6. Experimento 6: XGBoost

Finalmente, llegamos al último experimento en el que usaremos técnicas de *Boosting*, que a su vez es el experimento final de nuestro estudio. El paquete que utilizaremos en este caso se llama XGBoost, que ya fue explicado con más detalle en el punto 3.1.4 de este documento.

La función empleada para crear los modelos también se llama *xgboost*, y es bastante similar a la función *gbm* del experimento anterior. Las diferencias en cuanto al uso son que en *xgboost* hay que utilizar matrices en lugar de *data.frames* para manejar los datos y que hay que separar las entradas y las salidas, no podemos pasarle todos los datos juntos e indicar la fórmula en función de la salida como en el resto de experimentos.

En cuanto a la estructura de creación de modelos, el proceso es idéntico al resto de experimentos, empezando por tomar los primeros 75 atributos del fichero de datos de entrenamiento e incrementándolos de 75 en 75 en función de los puntos cercanos a la estación utilizados. La diferencia, como ya hemos dicho, es que en esta ocasión la salida no se añade a las columnas de los atributos, sino que se pasa a la función mediante otro parámetro. Además de los datos, debemos pasarle a la función los parámetros *max.depth, eta, nround* y *objective*, cuyos valores determinaremos en el siguiente apartado.

Una vez hayamos creado y entrenado los modelos, procederemos a realizar las predicciones de entrenamiento y test. Esta parte es idéntica a las predicciones del experimento anterior. Se calcularán con la función *predict*, a la que le pasaremos el modelo entrenado, los datos de test y el número de árboles utilizados.

Como en el resto de experimentos, al finalizar las ejecuciones deberemos tener 1568 modelos entrenados y 3136 predicciones calculadas.

4.2.6.1. Estudio de parámetros.

Los parámetros que vamos a utilizar en este caso son *max.depth, eta, nround* y *objective*. Este último es la función objetivo, y utilizaremos la regresión lineal, por lo que el valor para este parámetro será "*reg:linear*". Para ajustar el valor del resto de parámetros, realizaremos una búsqueda exhaustiva en rejilla con los conjuntos de entrenamiento y validación para cada combinación de dichos parámetros. Los rangos de los que partiremos para cada parámetros serán los siguientes:

eta = (0.1, 0.3, 0.6, 1).

max.depth = (2, 4, 6, 8).

nround = (1000, 5000, 7000, 10000).

Tras probar todas las combinaciones posibles con estos rangos de valores, el menor MAE de validación resultante es 2134124, y se obtiene con la combinación de parámetros *eta=0.1*, *max.depth=6* y *nround=1000*. Los valores ajustados para el número de árboles y su profundidad son valores intermedios de sus correspondientes rangos, pero la tasa de aprendizaje tiene el valor del extremo inferior de su rango, por lo que debemos probar con valores más pequeños de *eta* para ver si el error sigue mejorando. Incrementaremos el rango con los valores *eta = (0.01, 0.025, 0.05, 0.075)*.

Tras combinar los nuevos valores con el resto de rangos, conseguimos bajar el error hasta 2081449, que se obtiene con *eta=0.01*, *max.depth=6* y *nround=1000*. De nuevo, el menor error lo obtenemos con el extremo inferior del rango de valores de *eta*, por lo que debemos volver a ampliarlo con valores todavía más pequeños para comprobar si podemos seguir mejorando el error. Los nuevos valores serán *eta = (0.005, 0.0075)*.

Por tercera vez, conseguimos el error más bajo con el valor más pequeño del rango de valores de *eta*, que es 2051849, y lo obtenemos con *eta=0.005*, *max.depth=6* y *nround=1000*. Continuamos probando con tasas de aprendizaje inferiores. Los nuevos valores que probaremos serán *eta = (0.0001, 0.0005, 0.001, 0.0025)*.

Finalmente, tras incluir estos valores en nuestro estudio de parámetros, obtenemos el mejor resultado con *eta=0.0005*, *max.depth=6* y *nround=7000*, que proporcionan un MAE de validación de 2024389. Tras varias mejoras del error medio absoluto de validación conseguidas mediante la ampliación del rango de valores de *eta*, obtenemos una combinación de valores para los parámetros que nos proporciona el error más bajo del estudio sin estar ninguno de ellos en los extremos de sus correspondientes rangos, por lo que la combinación final resultante de este estudio de parámetros es *eta=0.0005*, *max.depth=6* y *nround=7000*.

4.2.6.2. Problemas surgidos durante la experimentación.

El estudio de los parámetros pudimos completarlo sin ningún problema, pero cuando pasamos a ejecutar las pruebas del experimento se producía un error del sistema de tipo APPCRASH en *rsession.exe* que paraba la ejecución del programa y no permitía

completar el experimento. En la Ilustración 5 que aparece a continuación, se muestra dicho error.

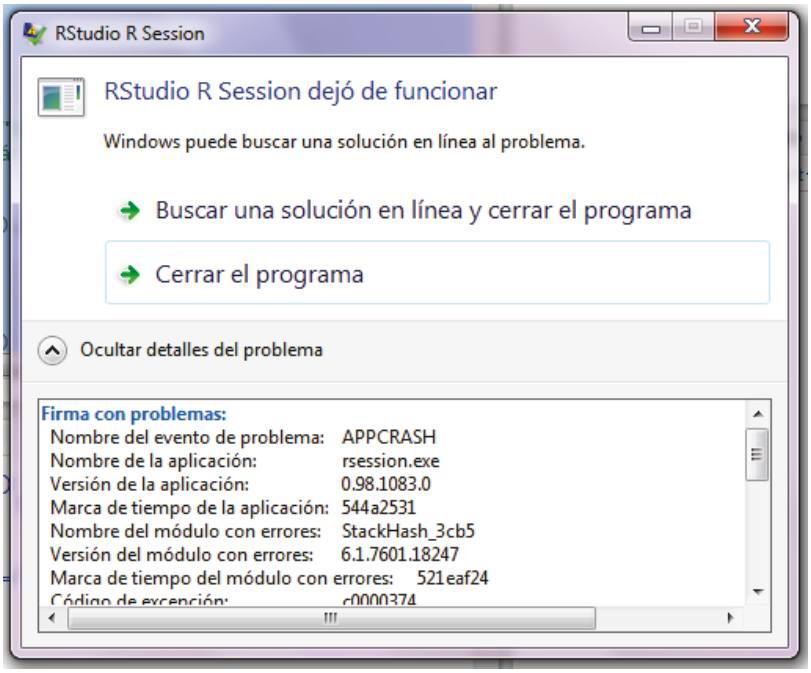


Ilustración 5. Error de tipo APPCRASH al ejecutar XGBoost.

El error parecía producirse de forma aleatoria, puesto que nunca se producía en el mismo punto de la ejecución. Al lanzar el script con la pruebas de la estación 1 por ejemplo, el error podía saltar en la prueba con los dos puntos más cercanos y al siguiente intento saltar en la prueba con nueve puntos cercanos. Tampoco se paraba siempre en el mismo tipo de sentencia de código, unas veces se paraba al hacer una predicción, otras al crear el modelo, otras al leer o escribir datos, e incluso alguna vez se paraba al crear una gráfica con la función *plot*. El código del APPCRASH tampoco era el mismo en todas las ocasiones.

Investigando este error, encontramos varias posibles causas y soluciones. Una posible solución era añadir el ejecutable que da el error (en este caso *rsession.exe*) como excepción en las configuraciones avanzadas del sistema, en la prevención de ejecución de datos (DEP)[44], pero no dio resultado. También comprobamos si podía ser un problema de versiones de R, pero el problema tampoco se producía por eso, ya que XGBoost está implementada con la versión 3.1.2, que es la que estábamos utilizando.

Intentamos también solucionarlo incrementando la memoria de ejecución de R, pero no sirvió. El sistema operativo que estabamos utilizado era Windows, así que decidimos probar con Linux, ya que es menos restrictivo con los recursos del sistema y maneja mejor la memoria. El problema persistía.

El ejecutable que provocaba el error, *rsession.exe*, pertenece a RStudio, así que probamos a utilizar directamente R sin la interfaz RStudio. El error seguía produciéndose, aunque el ejecutable cambiaba (*Rgui.exe*). Al fallar también el programa R sin RStudio, lo intentamos ejecutando R directamente desde la línea de comandos de Linux (*xterm*) sin usar ningún tipo de GUI, pero tampoco funcionaba.

Tras probar todo lo anterior, descubrimos que había dos versiones del paquete XGBoost: una versión estable, que era la que estábamos utilizando, y otra versión en desarrollo. Instalamos y probamos la versión en desarrollo, pero el error seguía produciéndose.

Buscando el error por internet, encontramos un caso muy similar en la web *Stackoverflow*[45] que llevaba meses abierto y aún no habían encontrado solución[46]. El error parecía complicado de solucionar, y ya nos había retrasado bastante intentar arreglarlo, así que tuvimos que tomar la decisión de cancelar el experimento con XGBoost.

Posteriormente, al acceder vía web al repositorio oficial de paquetes de R, en XGBoost[47] aparecía el mensaje que se muestra a continuación en la Ilustración 6:

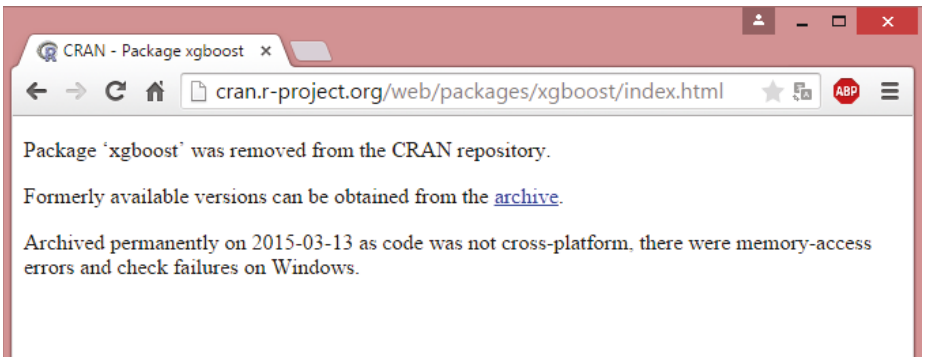


Ilustración 6. Mensaje sobre XGBoost en el repositorio de paquetes.

Esto podría indicar que el problema estaba en el propio paquete y por eso no éramos capaces de solucionar el problema.

Una vez explicados todos los experimentos realizados, pasamos a analizar y comparar los resultados obtenidos en cada uno de ellos.

4.3. Análisis y comparación de los resultados obtenidos.

Tras explicar en detalle los experimentos realizados y exponer los resultados obtenidos, en este apartado analizaremos y compararemos dichos resultados para poder llegar a conclusiones sobre los objetivos que planteamos al iniciar este proyecto.

Analizaremos principalmente la influencia de los parámetros en la precisión de cada uno de los métodos, los resultados de todos los métodos utilizados con sus mejores parámetros, la variabilidad del error por estación, la influencia del número de puntos del GEFS cercanos a las estaciones en cada método y los tiempos de ejecución de los experimentos.

4.3.1. Influencia de los parámetros en la precisión de los métodos.

Aquí veremos cómo influye cada parámetro de cada uno de los métodos en la precisión de los modelos creados. Únicamente hablaremos de *Gradient Boosting* y de las Máquinas de Vectores de Soporte, puesto que en el experimento con Regresión Lineal omitimos los parámetros durante las ejecuciones. El objetivo de este apartado es comprender mejor la sensibilidad de los distintos métodos a los valores de sus parámetros.

4.3.1.1. Parámetros de Gradient Boosting Regression.

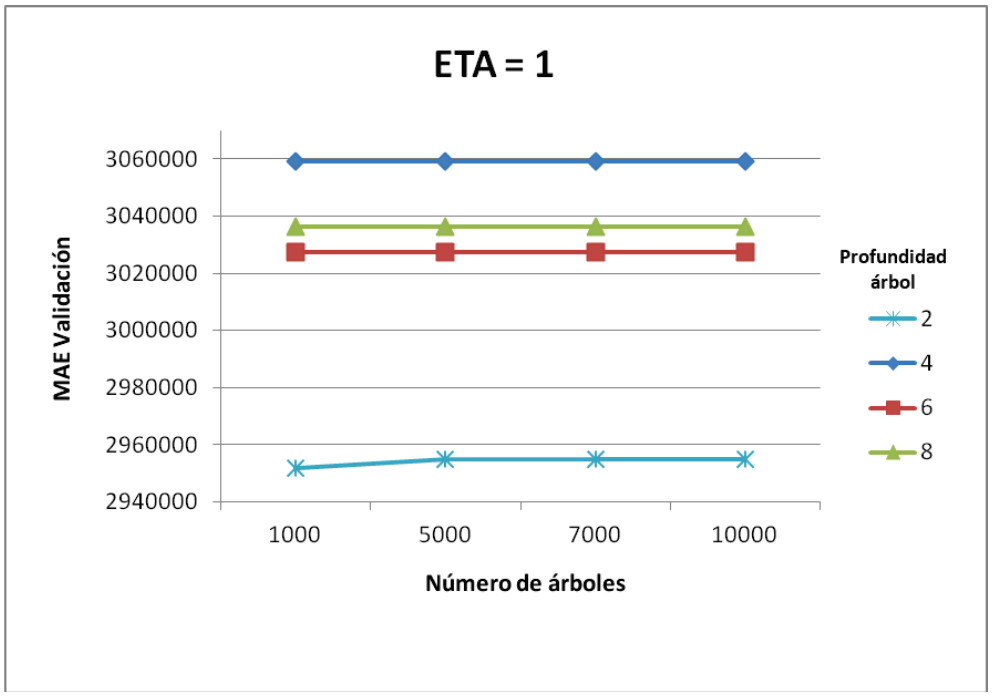
Realizamos dos experimentos con diferentes paquetes que implementaban *Gradient Boosting Regression*. Dichos paquetes fueron GBM y XGBoost. El experimento con GBM se pudo finalizar, pero el de XGBoost no, debido a los problemas explicados en el apartado 4.2.6.2 de este documento.

En el experimento con GBM no realizamos estudio de parámetros (consultar el apartado 4.2.5.1), pero en XGBoost sí pudimos realizar y completar el estudio de parámetros (apartado 4.2.6.1). Aunque no pudimos obtener resultados finales con XGBoost, podemos utilizar el estudio de parámetros para estudiar la influencia de cada parámetro en la precisión del modelo, puesto que el método de predicción utilizado sigue siendo *Gradient Boosting*, lo que cambia es la implementación del algoritmo, que hace que el entrenamiento sea más rápido.

En el experimento con XGBoost, los parámetros que utilizamos fueron la tasa de aprendizaje (ETA), el número de árboles y la profundidad de los mismos.

Primero, estudiaremos la influencia de los parámetros en función de la tasa de aprendizaje con varias gráficas para diferentes valores de dicho parámetro.

La Gráfica 13 representa el caso en el que el valor de ETA es la tasa de aprendizaje más alta que utilizamos en el estudio de parámetros.

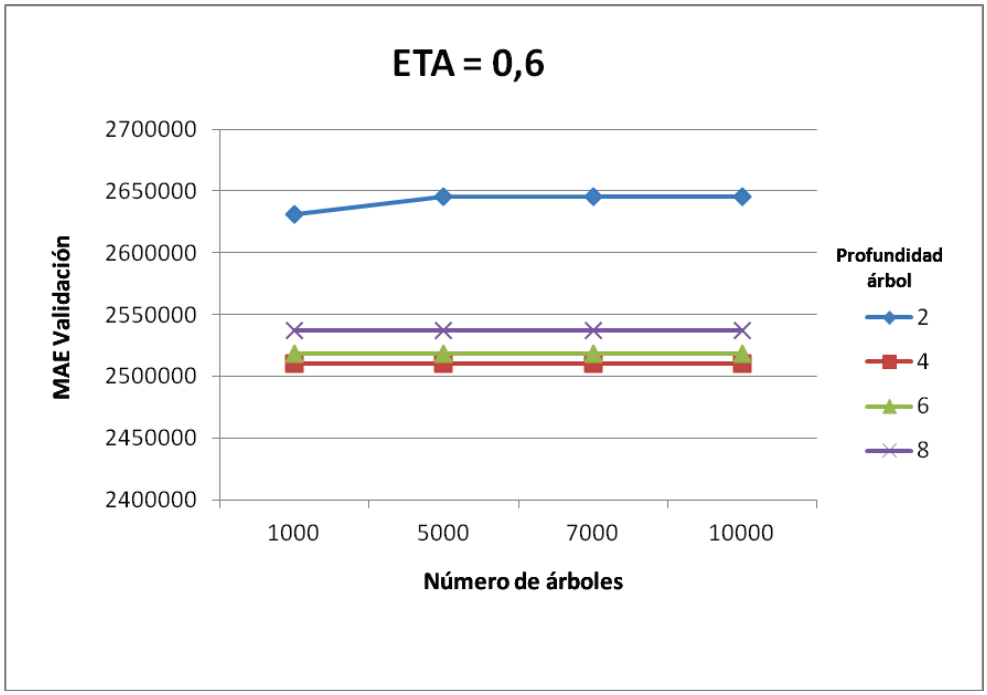


Gráfica 13. Influencia de los parámetros de XGBoost cuando ETA=1.

Con una tasa de aprendizaje de 1, se puede observar que el MAE apenas varía al aumentar el número de árboles. Esto se debe a que con una tasa de aprendizaje alta, el árbol aprende más rápido, pero menos preciso.

También observamos que con menos profundidad por árbol, obtenemos mejores resultados. Los peores resultados no se obtienen con la mayor profundidad probada (8), se obtienen con profundidad 4, pero la diferencia entre ellos no es muy significativa.

En las siguientes gráficas (Gráficas 14, 15 y 16), representamos lo mismo pero con valores de ETA intermedios en el rango que utilizamos para el estudio.

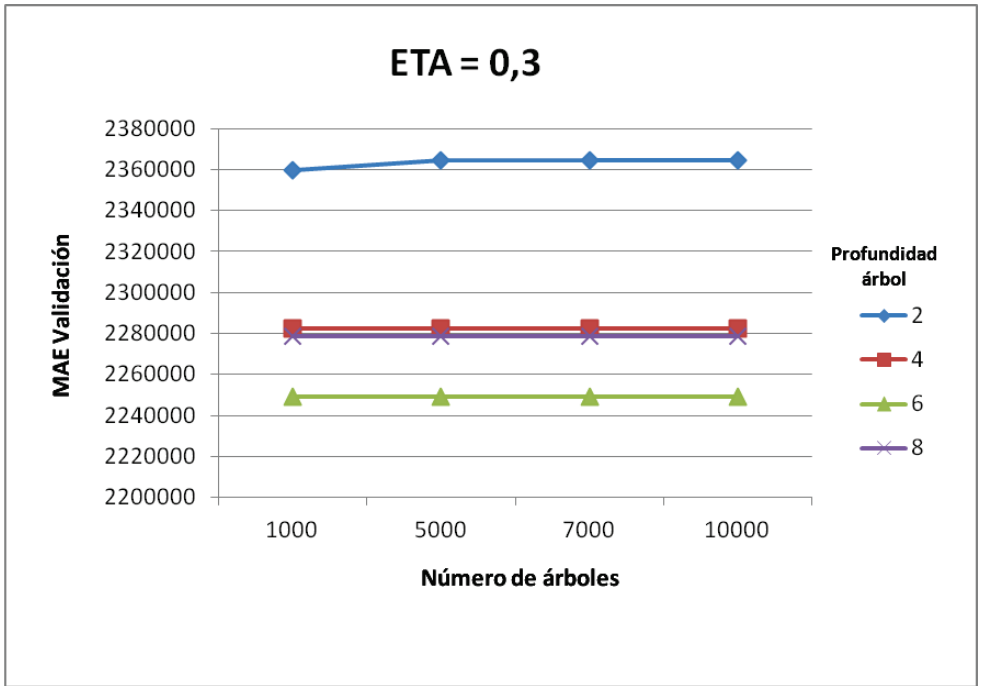


Gráfica 14. Influencia de los parámetros de XGBoost cuando ETA=0,6.

Bajando la tasa de aprendizaje hasta 0.6, el error mejora bastante con respecto al caso anterior (*ETA=1*). Esto se debe a que el modelo aprende mejor cuando la tasa de aprendizaje es más baja, aunque tarde más.

Respecto a la influencia del número de árboles, este caso es prácticamente igual que el anterior. El error únicamente varía de manera visible cuando la profundidad es 2, y no mejora con el incremento del número de árboles. La causa más probable es que con esa tasa de aprendizaje, el modelo tarda menos en aprender.

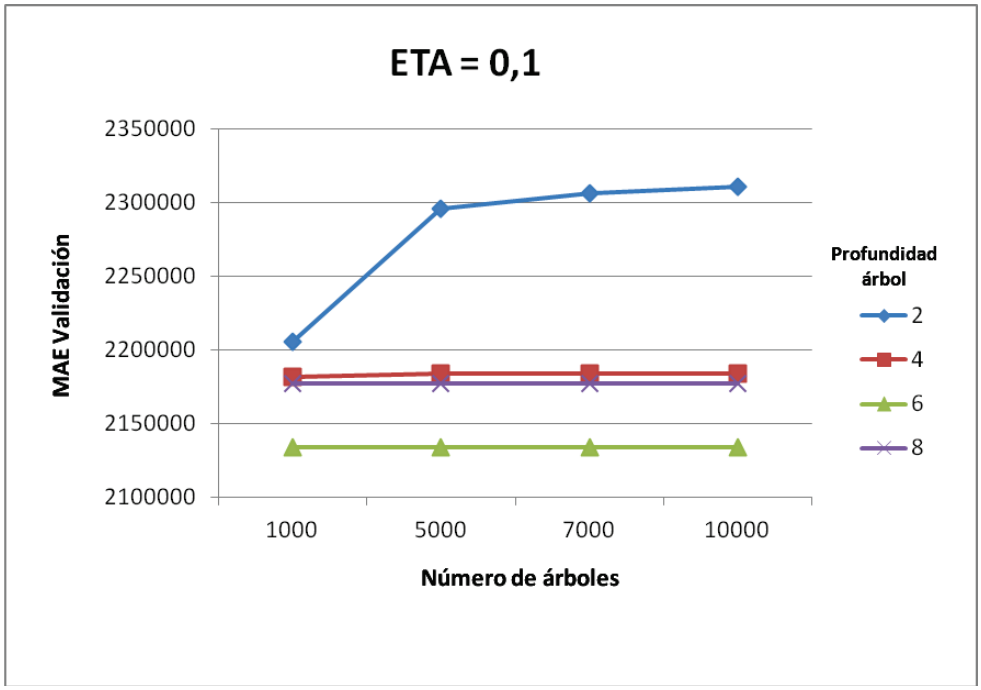
La influencia de la profundidad de los árboles sí que cambia respecto al caso anterior. Ésta vez el error sí mejora cuando la profundidad aumenta. Esto se debe también a la tasa de aprendizaje, que como ya no es tan alta como en el caso anterior, tarda más tiempo en aprender. El mejor resultado se obtiene con profundidad 4, aunque es prácticamente igual con profundidades de 6 y 8.



Gráfica 15. Influencia de los parámetros de XGBoost cuando ETA=0,3.

Bajando la tasa de aprendizaje nuevamente (hasta 0.3), el error vuelve a mejorar (como venimos diciendo, con tasas de aprendizaje más bajas, el modelo aprende más lento pero más preciso).

En cuanto a la influencia del resto de parámetros en el error, la situación es muy similar a la gráfica anterior. El incremento del número de árboles sigue influyendo poco, aunque parece que la profundidad va influyendo más (ésta vez, el mejor resultado se obtiene con profundidad 6).

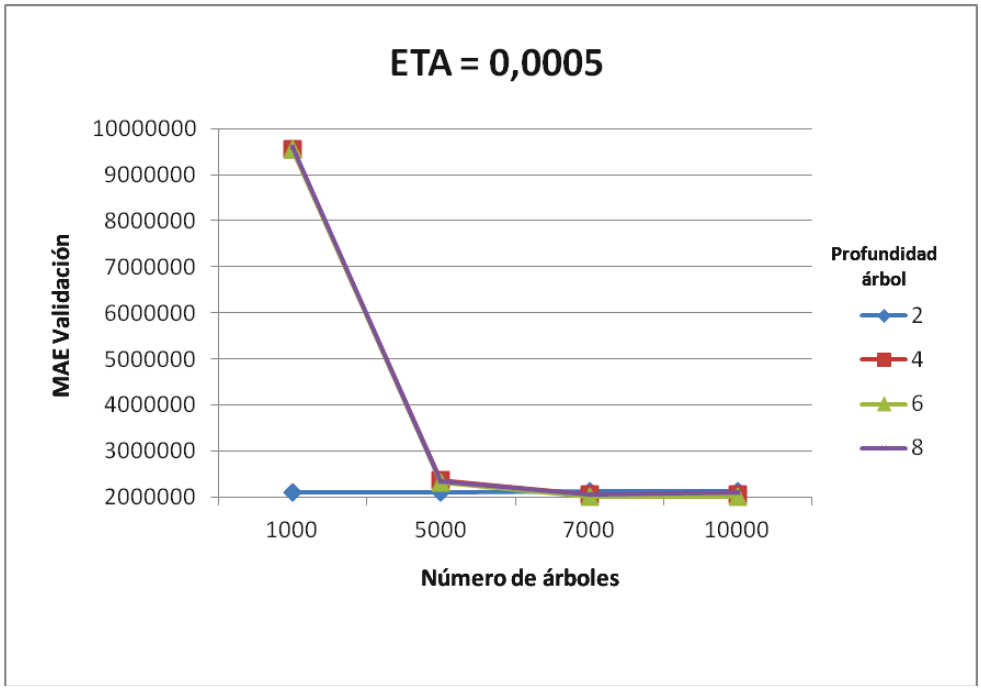


Gráfica 16. Influencia de los parámetros de XGBoost cuando ETA=0,1.

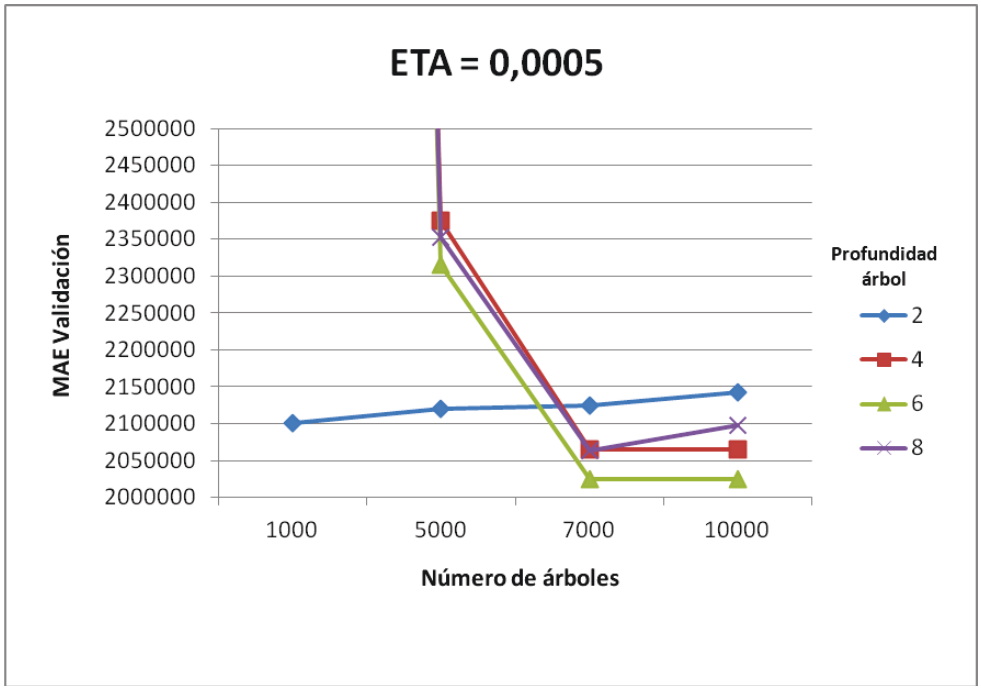
Bajando de nuevo el valor de ETA, esta vez hasta 0.1, se observa que varía algo más el MAE al aumentar el número de árboles (con profundidad 2, la variación es más significativa; y con profundidad 4, dicha variación es muy pequeña pero ya se empieza a apreciar, a diferencia de los casos anteriores), pero sigue siendo una variación a peor.
Los MAE resultantes vuelven a mejorar al haber disminuido nuevamente la tasa de aprendizaje.

La influencia del parámetro de profundidad es prácticamente igual que en la gráfica anterior.

Por último, en las siguientes gráficas representamos la influencia de los parámetros para la tasa de aprendizaje que nos proporciona los mejores resultados, que es *ETA=0,0005*. Ambas gráficas (Gráficas 17 y 18) representan la misma situación, la Gráfica 17 a escala completa para ver la forma de las curvas, y la Gráfica 18 con zoom en la escala para poder ver con detalle el comportamiento de dichas curvas.



Gráfica 17. Influencia de los parámetros de XGBoost cuando ETA=0,0005.

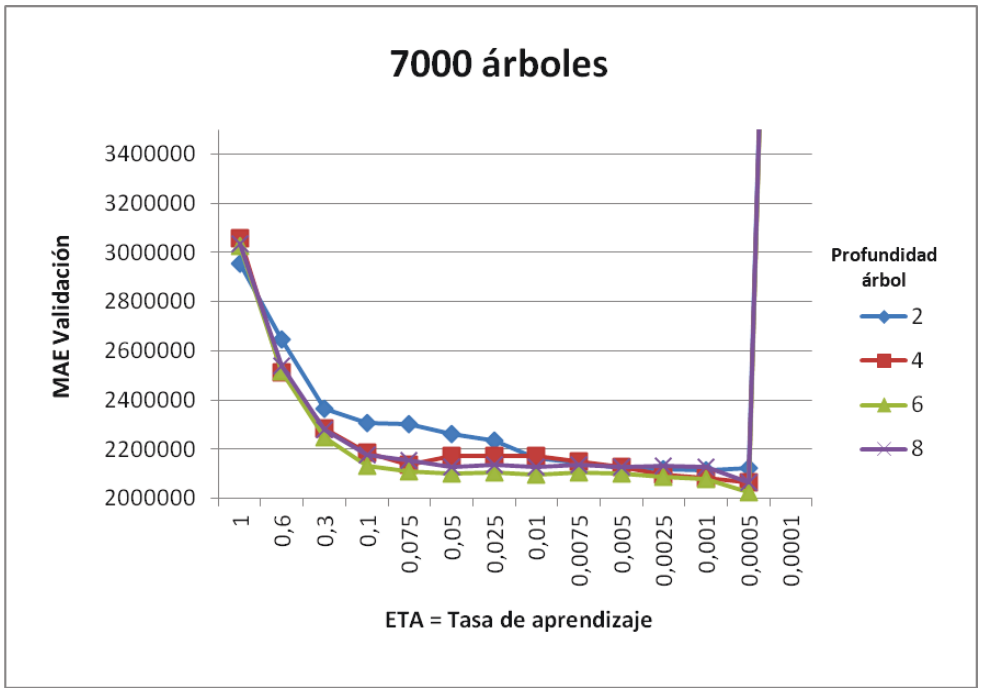


Gráfica 18. Influencia de los parámetros de XGBoost cuando ETA=0,0005 con zoom en la escala.

Con valores más pequeños de ETA, vemos que el comportamiento cambia. En la gráfica podemos ver que, en este caso, el error tiende a

mejorar a mayor profundidad y al aumentar el número de árboles. Aquí ya observamos mucho mejor que con una tasa de aprendizaje tan pequeña, el modelo tarda bastante más en aprender. El mejor resultado se obtiene con 7000 árboles de profundidad máxima 6. Con 10000 árboles y esa misma profundidad, el error es el mismo, por lo que no merece la pena aumentar el número de árboles a más de 7000 porque el aprendizaje se estanca. Esto no contradice el hecho de que con más árboles y menor tasa de aprendizaje, el modelo aprende mejor. La idea es esa, lo que hay que encontrar son los parámetros adecuados a nuestros datos. Si nos pasamos, el error no mejora. Aumentando la profundidad de los árboles hasta 8, el error empeora levemente, aunque no hay mucha diferencia con respecto a profundidad 6 o 4.

Ahora, pasaremos a estudiar la influencia de los parámetros en función del número de árboles utilizados en el ajuste. La Gráfica 19 representa la influencia de la tasa de aprendizaje y la profundidad en el error para el mejor número de árboles de nuestro estudio de parámetros, que es 7000.



Gráfica 19. Influencia de los parámetros de XGBoost para el caso con 7000 árboles.

En la gráfica podemos observar claramente la tendencia a la baja que tiene el error absoluto medio cuando la tasa de aprendizaje va

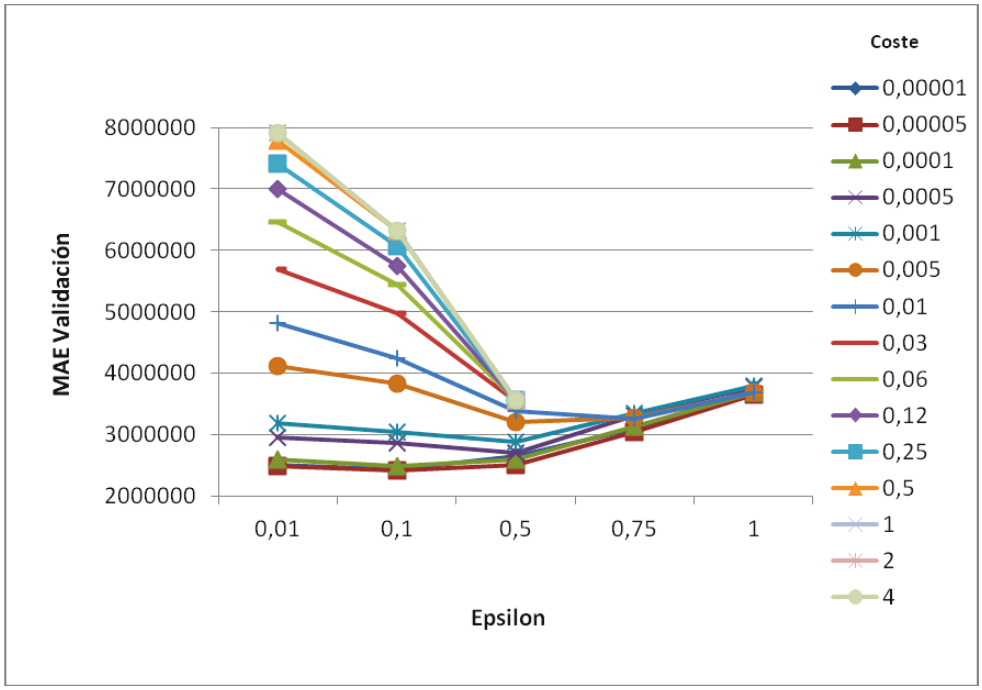
disminuyendo. Aunque para el valor más bajo de ETA, vemos que el error se dispara. Eso se debe a que esa tasa de aprendizaje es demasiado pequeña como para que el modelo converja en una solución óptima.

También podemos ver que el error tiende a mejorar cuando aumenta la profundidad de los árboles. Los mejores resultados se obtienen con profundidad 6, aunque la mayor probada es 8, pero los resultados con dicha profundidad son bastante parecidos. Esto indica que la profundidad adecuada es 6, y que con profundidad de 8 en adelante, los árboles serían demasiado grandes y los resultados del modelo empeorarían.

Lo que observamos en esta gráfica, nos ayuda a confirmar lo observado sobre la influencia de los parámetros en el método XGBoost en las gráficas anteriores.

4.3.1.2. Parámetros de las SVM con kernel polinómico.

Ahora estudiaremos la influencia de los parámetros de las Máquinas de Vectores de Soporte utilizando kernel polinómico. Los parámetros utilizados en éste método son el *Coste* y *Epsilon*.



Gráfica 20. Influencia de los parámetros en la precisión de las SVM con kernel polinómico.

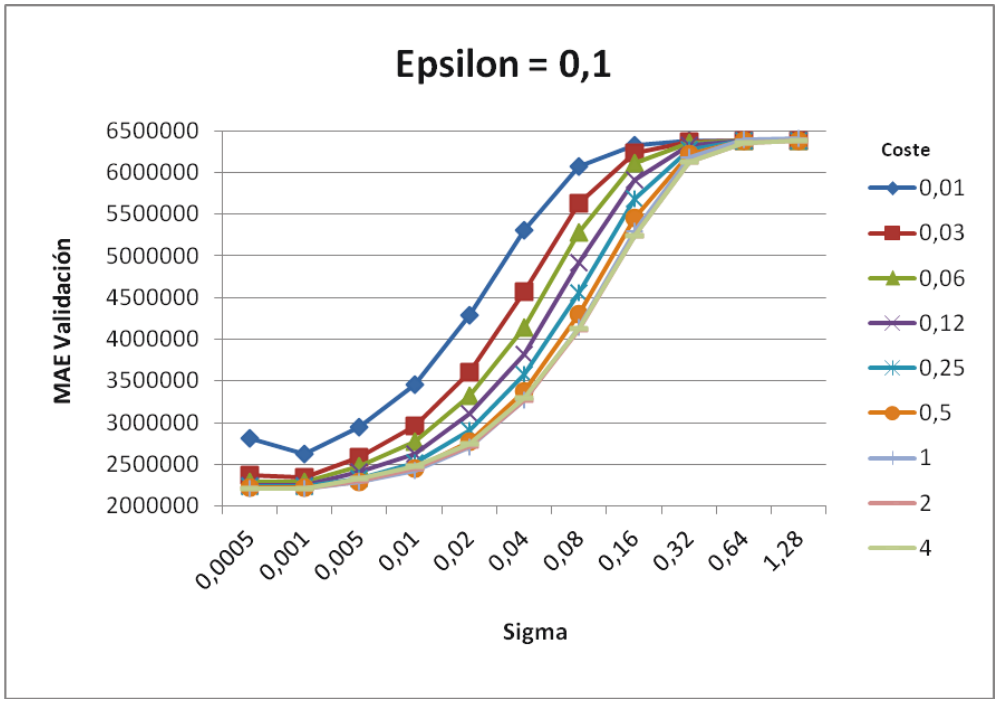
Ésta gráfica (Gráfica 20) representa los resultados del error de validación en el estudio de parámetros para el método SVM con kernel polinómico. Cada curva representa los resultados del error absoluto medio con un *coste* determinado, en función de los diferentes valores de *epsilon*.

Lo primero que nos muestra la gráfica es que, a mayores costes, mayor es el MAE obtenido y, por consiguiente, con costes más bajos, el error es menor. En las curvas de los costes más altos, podemos observar que el error tiende a disminuir considerablemente según aumenta el valor de *epsilon* hasta los valores intermedios de nuestra escala. Después, la mejora se estanca. Sin embargo, en las curvas de los costes más bajos, vemos que el error tiende a aumentar según se incrementa el valor de *epsilon*. Podemos decir por tanto, que a menor *coste* y menor valor de *epsilon*, los resultados tienden a ser mejores. El mejor resultado se obtiene con *coste* 0.00005 y *epsilon* 0.1.

4.3.1.3. Parámetros de las SVM con kernel Gaussiano.

Por último, analizaremos la influencia de los parámetros en el error de las Máquinas de Vectores de Soporte con kernel Gaussiano. Los parámetros utilizados en éste método fueron *C (coste)*, *epsilon* y *sigma*.

En la Gráfica 21 se representa la influencia de los parámetros *sigma* y *coste* en el estudio de parámetros para el mejor valor de *epsilon*, que es 0.1. En esta gráfica, cada curva representa un valor para el parámetro *C (coste)*.



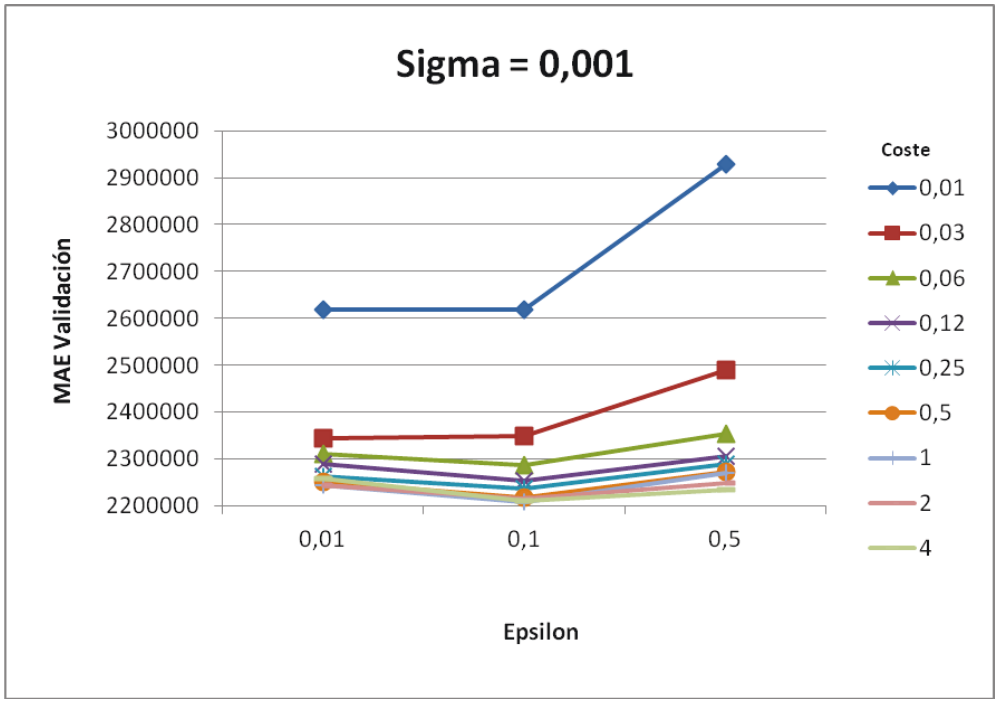
Gráfica 21. Influencia de los parámetros Sigma y Coste en SVM con kernel Gaussiano.

Como podemos observar en las curvas de esta gráfica, los resultados mejoran cuando se va incrementando el valor del *coste* (a diferencia de lo que ocurría en la gráfica de las SVM con kernel polinómico).
En cuanto a la influencia del parámetro *sigma*, podemos ver que el error tiende a aumentar según se incrementa el valor de *sigma*.

Por tanto, podemos decir que a mayor *coste* y menor valor de *sigma*, los resultados tienden a ser mejores.

El mejor error absoluto medio de este estudio de parámetros lo hemos obtenido con valor de *sigma* 0.001 y un *coste* de 1.

En la segunda gráfica de este apartado (Gráfica 22), representamos la influencia de los parámetros *coste* y *epsilon* en los resultados para el mejor valor de *sigma*, que según nuestro estudio es 0.001.



Gráfica 22. Influencia de los parámetros Epsilon y Coste en SVM con kernel Gaussiano.

Lo primero que podemos ver en las curvas que, al igual que en la gráfica anterior, cada una representa un valor para el parámetro *coste*, el error absoluto medio tiende a disminuir al aumentar el *coste*, como ya veíamos en la gráfica anterior.

En cuanto al parámetro *epsilon*, podemos observar que en todas las curvas, el error mejora (o no empeora) cuando *epsilon* aumenta de 0.01 a 0.1; pero cuando se sigue incrementando su valor, el error empeora.

Esto encaja con lo que pudimos interpretar en la gráfica anterior.

4.3.2. Resultados de todos los métodos utilizados con sus mejores parámetros.

En este apartado analizaremos y compararemos los resultados obtenidos en todos los métodos con sus mejores parámetros. Dado que la cantidad total de datos es muy grande (3136 resultados por cada uno de los 4 métodos), calcularemos la media de las 98 estaciones para trabajar con ella. También calcularemos la desviación típica.

La Tabla 10 recoge la media de los resultados de las 98 estaciones para cada método y número de puntos del GEFS cercanos utilizados para crear el modelo.

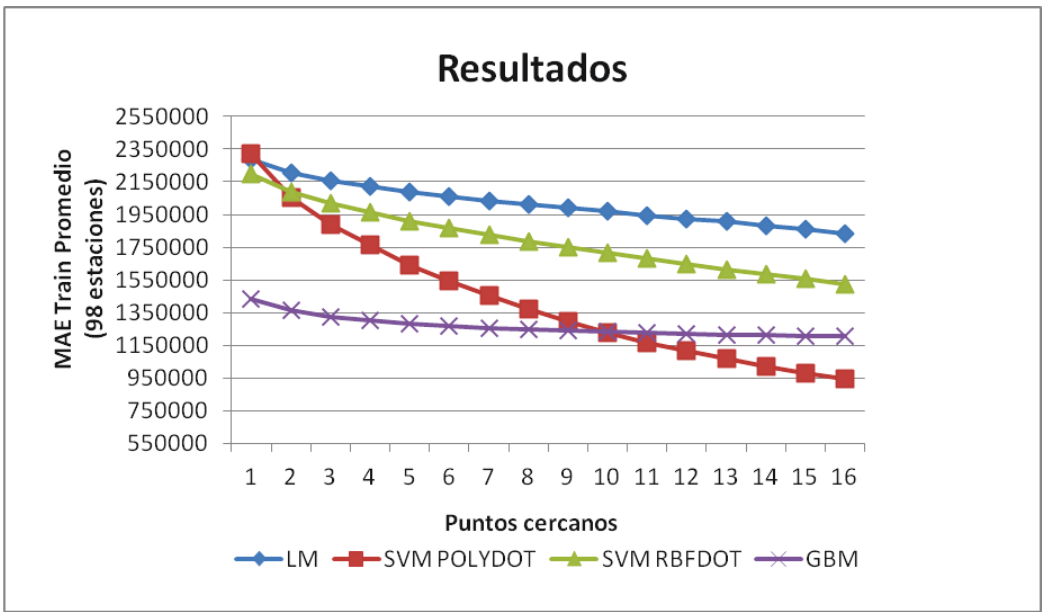
MEDIA ARITMÉTICA DEL MAE DE LAS 98 ESTACIONES PARA CADA CASO								
Puntos cercanos	LM		SVM POLYDOT		SVM RBFDOT		GBM	
	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST
1	2289589,82	2226545,94	2321784,58	2263055,27	2196550,87	2080477,52	1433200,9	1997768,6
2	2205927,54	2196503,69	2050742,93	2141330,05	2085322,09	2012230,18	1361301,61	1966513,37
3	2155292,63	2199000,66	1886149,2	2142795,49	2016145,87	1984942,75	1324578,65	1950992,64
4	2123212,89	2225742,41	1764662,19	2181572,53	1964489,48	1976414,51	1301536,5	1944217,23
5	2085323,99	2241261,39	1641073,04	2222744,5	1907660,56	1965965,04	1284744,22	1934425,93
6	2060647,59	2274730	1543617,22	2286196,27	1865242,3	1967298,56	1266364,46	1932713,33
7	2034913,27	2307471,56	1453658,57	2355101,9	1824087,23	1969488,25	1255941,24	1930188,59
8	2011922,71	2340029,53	1372274,55	2427752,25	1785997,48	1973412,02	1246510,74	1929248,28
9	1992059,11	2374271,29	1298665,08	2502964,91	1748731,41	1979743,26	1238488,93	1927967,67
10	1967797,59	2411634,95	1229816,15	2578013,71	1713036,1	1986219,95	1231161,67	1926007,09
11	1945203,74	2445595,17	1167567,06	2638755,12	1677991,11	1991857,77	1224003,93	1924637,72
12	1923462,26	2475462,85	1113231,1	2701952,38	1645416,98	1997688,29	1219801,74	1924914,71
13	1907094,74	2518148,03	1064772,82	2774397,83	1614970,32	2006401,94	1214928,31	1925338,69
14	1879688,12	2563143,89	1021052,68	2834152,89	1584838,75	2014062,92	1209771,55	1923344,02
15	1857045,63	2607515,01	980007,136	2895190,27	1554511,15	2020792,79	1206580,92	1916697,38
16	1834355,54	2645270,15	944514,366	2954773,54	1525627,42	2029916,26	1203861,02	1924423,4

Tabla 10. Media de los resultados de las 98 estaciones para cada método y combinación de puntos cercanos utilizados.

En la tabla podemos ver que el método más preciso es GBM, ya que es el método que nos proporciona el menor error absoluto medio de test. Las Máquinas de Vectores de Soporte con kernel Gaussiano obtienen resultados algo peores que GBM pero muy parecidos.

Teniendo en cuenta los resultados medio, que son los representados en esta tabla, LM es mejor que SVM con kernel polinómico, puesto que de media obtiene predicciones más precisas en todos los casos menos cuando se utilizan entre 2 y 5 puntos cercanos para entrenar los modelos, que son más precisas las SVM. No obstante, si tenemos en cuenta los mejores resultados por estación, que fueron expuestos en los apartados 4.2.1.4 y 4.2.3.4 (Tablas 3 y 7 respectivamente), las SVM con kernel polinómico obtienen mejor resultado que LM en 77 de las 98 estaciones, un 78,5% aproximadamente, por lo que fijándonos en esos datos SVM Polydot sería mejor que LM. Podríamos decir que con menos datos de entrada, las SVM con kernel polinómico son mejor método que LM, pero cuando se utilizan más datos de entrada, LM es mejor para realizar predicciones que SVM con kernel polinómico.

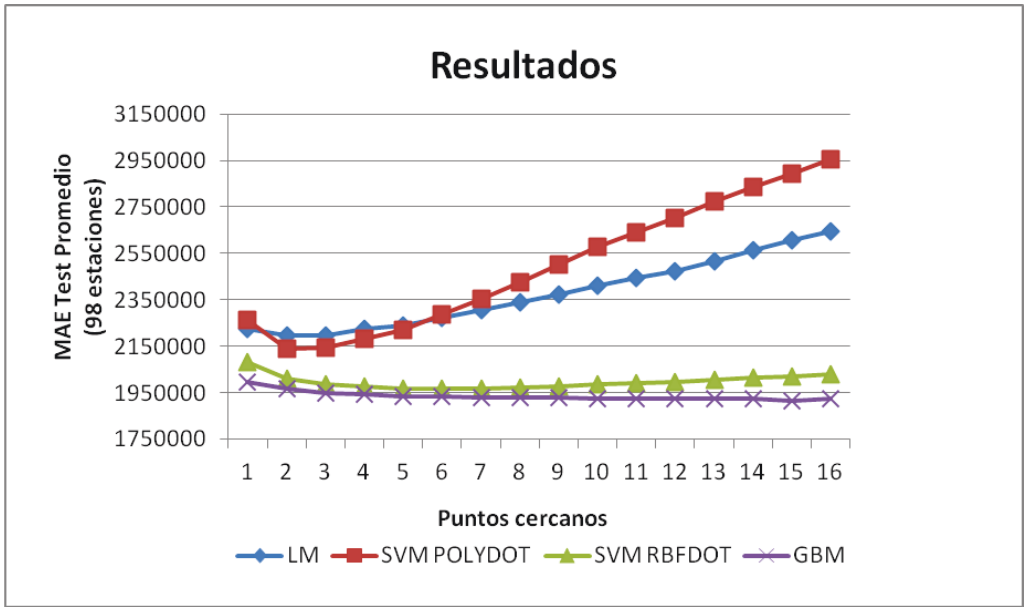
Ahora analizaremos la evolución de los errores de entrenamiento y test de los cuatro métodos.



Gráfica 23. Evolución de los MAE de entrenamiento de cada método.

En ésta gráfica se representan los errores absolutos medios de entrenamiento obtenidos por cada método de predicción, en función de la cantidad de datos de entrada utilizados (número de puntos más cercanos a cada estación utilizados). Los MAE representados en las curvas son el promedio de las 98 estaciones.

Podemos ver que en todos los métodos, el error absoluto medio de entrenamiento tiende a bajar según aumenta la cantidad de atributos de entrada. La explicación a esto es que cuantos más atributos de entrada utilizamos para entrenar los modelos, éstos son más complejos y tienen más grados de libertad para ajustarse a los valores a predecir. Esto no siempre es bueno, pues podría ocurrir que al utilizar datos nuevos, el modelo no hiciese predicciones muy precisas, porque al aprenderse unos datos concretos no podría generalizar (es decir, podría haber sobreaprendizaje).



Gráfica 24. Evolución de los MAE de test de cada método.

En la Gráfica 24, representamos los errores absolutos medios de test obtenidos por cada método de predicción, en función de la cantidad de datos de entrada utilizados (número de puntos más cercanos a cada estación utilizados). Los MAE representados en las curvas también son el promedio de las 98 estaciones, como en el caso anterior.

Las curvas de LM y SVM con kernel polinómico son bastante parecidas. En ellas, observamos que los resultados tienden a empeorar al aumentar la cantidad de atributos de entrada. Comparando ambas curvas, también podemos decir que, con menos datos, el modelo lineal consigue peores resultados que las SVM con kernel polinómico, aunque cuando el volumen de datos de entrada es mayor, las predicciones de las SVM son bastante menos precisas.

Por otro lado, observando las curvas de SVM con kernel Gaussiano y de GBM, también son bastante parecidas entre sí. En la curva de SVM con kernel Gaussiano, vemos que el error tiende a mejorar ligeramente al aumentar la cantidad de datos de entrada. El error varía bastante poco entre los puntos 5 y 8, y al continuar aumentando los datos de entrada, los resultados comienzan a empeorar. Podríamos decir que a partir de ese punto se produce sobreaprendizaje, puesto que el error de entrenamiento es cada vez menor pero el de test comienza a aumentar (el modelo generaliza peor). En la de GBM sin embargo, vemos que el error más bajo se obtiene con muchos datos (utilizando como entrada los datos de 15 de los 16 puntos más cercanos a las estaciones), pero la curva se ve bastante constante desde el punto 5, porque el error varía

bastante poco. Del punto 15 al 16 el error empeora, pero la variación es muy pequeña.

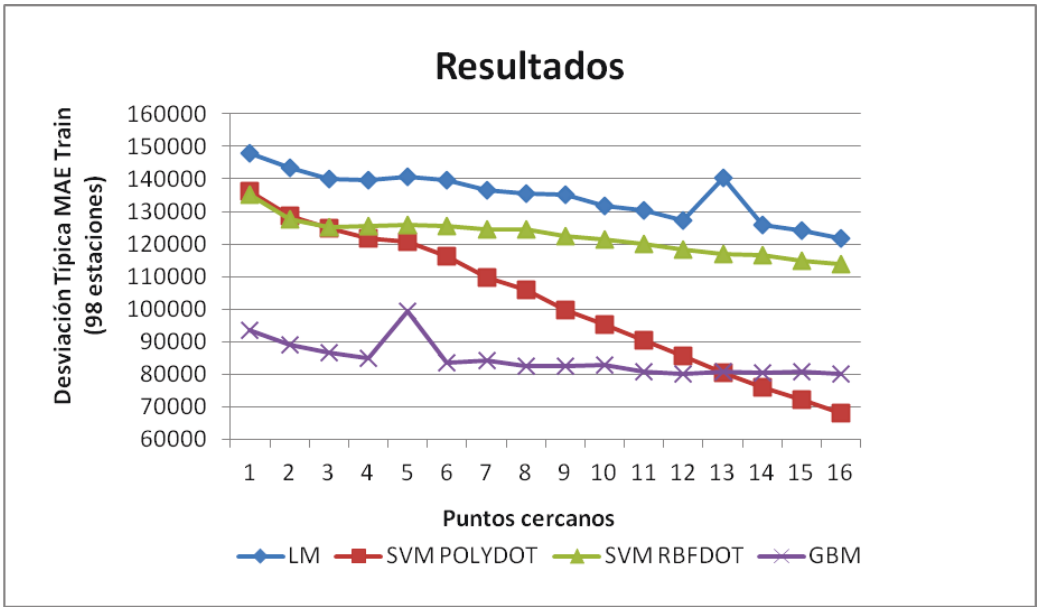
Los mejores resultados los obtenemos con GBM, pero los resultados de las SVM con kernel gaussiano son muy parecidos, y dada la gran diferencia de recursos necesarios para ejecutar un método y otro, podría considerarse más adecuado utilizar las SVM con kernel Gaussiano, sacrificando un poco de precisión en los resultados pero ahorrando recursos y tiempo de cómputo.

DESVIACIÓN TÍPICA DEL MAE DE LAS 98 ESTACIONES PARA CADA CASO								
Puntos cercan.	LM		SVM POLYDOT		SVM RBFDOT		GBM	
	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST
1	147747,816	191195,343	136240,373	189734,613	135077,01	182517,635	93544,9262	191598,165
2	143481,653	185998,673	128731,21	186328,821	127692,286	185163,158	88910,3307	192354,274
3	140129,053	183654,079	124706,027	178081,357	125264,714	182674,188	86651,8845	188944,865
4	139704,977	181845,134	121783,579	180702,305	125353,68	183411,367	84996,4613	188651,235
5	140557,76	183041,809	120678,455	173848,428	125913,4	183593,334	99168,5863	188987,496
6	139525,072	182264,734	116317,958	173570,564	125627,261	183937,717	83528,7538	186296,423
7	136671,245	183862,695	109802,14	175996,384	124438,512	183755,857	84152,9097	185630,877
8	135345,08	181193,051	105887,382	179689,771	124363,173	182698,113	82577,5218	185347,118
9	135089,632	183772,658	99639,4744	185532,696	122551,908	183243,615	82438,2274	188143,882
10	131844,213	185980,448	95233,2961	182010,262	121294,28	183590,523	82924,3345	187275,409
11	130268,511	188261,008	90320,293	181591,442	119931,564	183614,036	80812,5625	187797,584
12	127368,182	189209,368	85605,0768	191261,024	118249,437	183271,53	80012,8128	187373,83
13	140437,778	192981,655	80285,4148	187597,508	116987,913	183034,196	80746,6948	188367,457
14	125710,282	195722,371	75850,1871	199577,57	116407,283	181832,552	80470,384	187232,487
15	124197,106	198021,804	71955,3011	200417,189	114952,213	180540,334	80574,3893	197575,631
16	121593,357	210604,908	68039,2739	210551,968	113811,44	179023,731	80048,1262	189987,148

Tabla 11. Desviación típica de los resultados de las 98 estaciones para cada método y combinación de puntos cercanos utilizados.

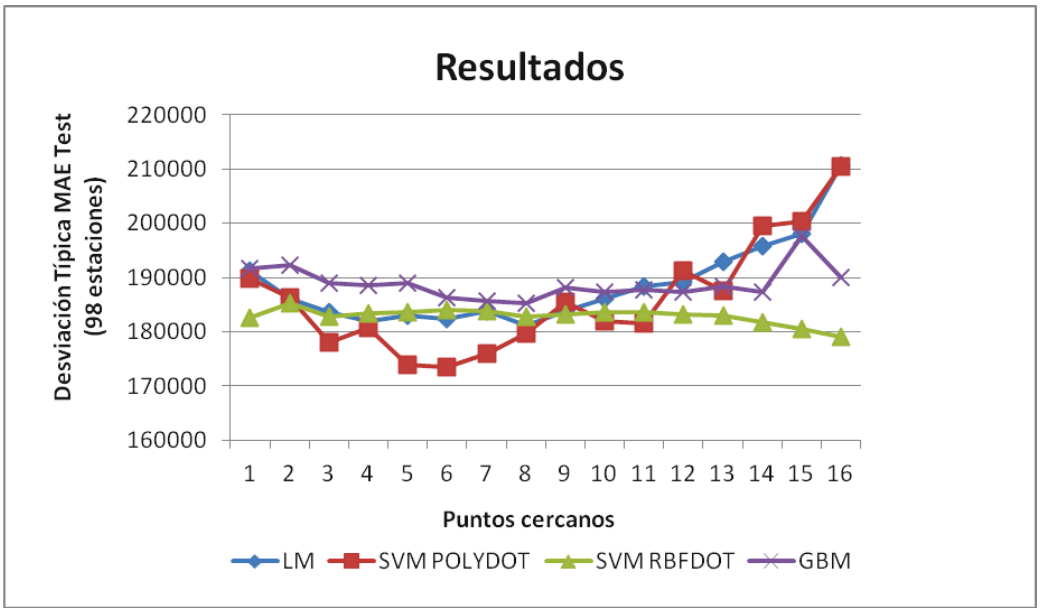
Con la desviación típica, medimos cuánto tienden a alejarse los valores concretos de la media. En nuestro estudio, la desviación típica de test de los cuatro métodos se mueve por valores similares. La desviación menor se da en las Máquinas de Vectores de Soporte, con kernel Gaussiano en caso de usar 1, 2, 9, 12, 13, 14, 15 y 16 puntos cercanos, y con kernel polinómico en el resto de modelos.

En la Gráfica 25 podemos ver las curvas que representan la desviación típica de entrenamiento de cada método. Vemos que en todos los métodos, la desviación típica tiende a disminuir cuando se incrementa el número de puntos cercanos utilizados para entrenar los modelos.



Gráfica 25. Desviación típica de los MAE de entrenamiento de cada método.

En las curvas de LM y GBM se presenta un pico en los puntos 13 y 5 respectivamente. Dado el comportamiento de las curvas antes y después de dichos picos, podemos considerarlos como ruido. A continuación, representaremos la desviación típica del error de test en la Gráfica 26.

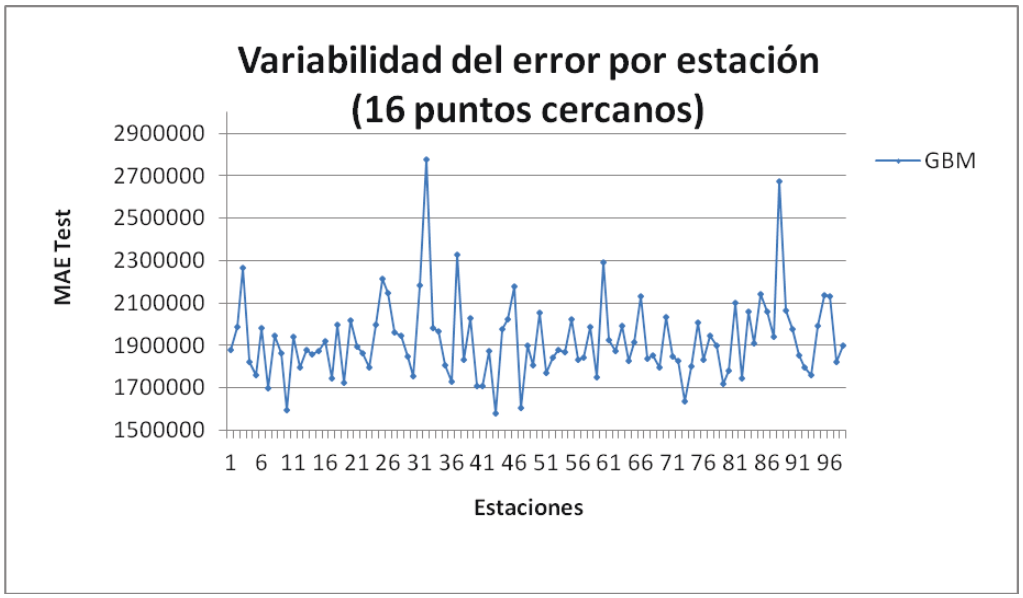


Gráfica 26. Desviación típica de los MAE de test de cada método.

En esta otra gráfica, vemos que en LM y SVM Polydot los puntos de la curva de la desviación típica de test están más dispersos que los de GBM y SVM Rbfdot. Eso se debe a que las variaciones entre modelos entrenados con diferentes puntos cercanos son más dispares.

4.3.3. Variabilidad del error por estación.

En este punto, analizaremos la variabilidad del error absoluto medio obtenido para cada estación cuando se han usado todos los puntos cercanos a las estaciones de nuestros conjuntos de datos de entrenamiento.



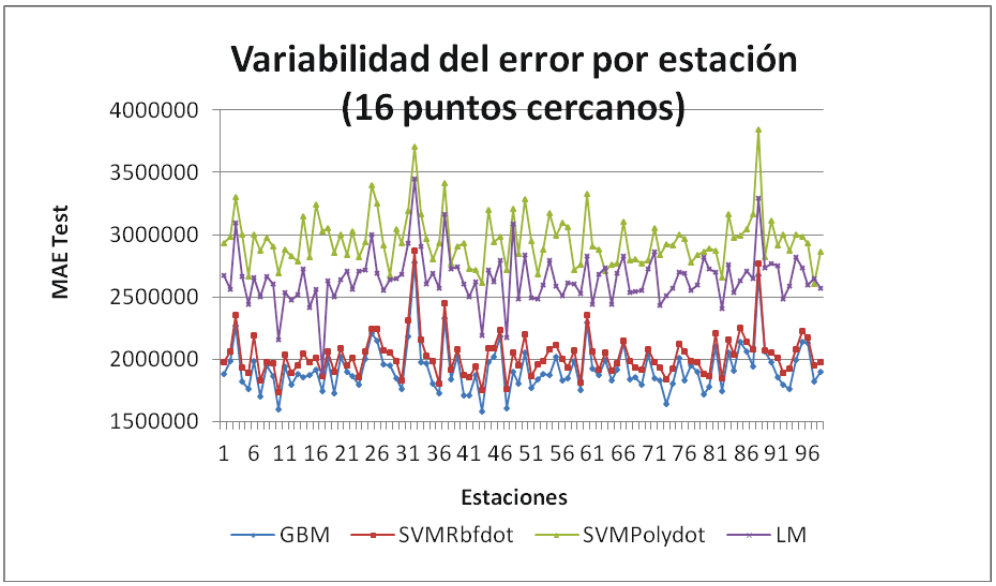
Gráfica 27. Variabilidad del error por estación al utilizar 16 puntos cercanos con el mejor método del estudio.

Ésta gráfica representa la variabilidad del error absoluto medio por estación del método de predicción más preciso de todos los empleados en el estudio (GBM) con sus mejores parámetros, utilizando los datos de entrada de los 16 puntos más cercanos a cada estación.

Podemos apreciar algunos picos en la gráfica que sobresalen más que otros. Esto se produce porque hay estaciones para las que el algoritmo GBM es bastante menos preciso que para el resto de estaciones. Sería interesante analizar esto para averiguar qué es lo que provoca dicha variabilidad.

La siguiente gráfica también representa la variabilidad del error absoluto medio por estación utilizando los datos de entrada de los 16 puntos más cercanos a cada de ellas, pero presenta una curva por cada método de

predicción utilizado en el estudio, no sólo el más preciso, cada uno con sus mejores parámetros.



Gráfica 28. Variabilidad del error por estación al utilizar 16 puntos cercanos con todos los métodos.

Podemos ver claramente que, de los cuatro métodos con los que hemos finalizado el estudio, dos de ellos (GBM y SVM con kernel Gaussiano) son bastante mejores que los otros dos (SVM con kernel polinómico y LM).

Fijándonos únicamente en los dos mejores, podemos ver que los resultados son muy parecidos, siendo el error absoluto medio de las SVM con kernel Gaussiano ligeramente peor que el error de GBM. Aunque como ya hemos comentado anteriormente, teniendo en cuenta los tiempos de ejecución de cada método, podría ser mucho más rentable utilizar las SVM con kernel Gaussiano que GBM. Tendríamos que sacrificar algo de precisión, pero a cambio ganaríamos mucho en tiempo de cómputo y recursos.

Si observamos las cuatro curvas, sin tener en cuenta que tengan mejor o peor error, podemos ver que son bastante parecidas, pues los picos que destacábamos en la gráfica anterior se presentan siempre en las mismas estaciones, independientemente del método de predicción. Esto nos puede llevar a deducir que las variaciones tan grandes del error entre unas estaciones y otras se pueden deber a circunstancias ajenas a los métodos de predicción, como podrían ser, por ejemplo, cuestiones geográficas de las estaciones. Sería interesante poder estudiar este tema.

4.3.4. Balance del número de puntos del GEFS cercanos a las estaciones en cada método.

En este apartado, estudiaremos cuáles son las cantidades de puntos cercanos a las estaciones más eficientes para obtener la mayor precisión de cada uno de los métodos utilizados.

En la Tabla 12, representamos la cantidad de estaciones que obtienen su mejor resultado utilizando las diferentes cantidades de puntos del GEFS cercanos por cada método de predicción utilizado.

Puntos cercanos	LM Estaciones	SVM Polydot Estaciones	SVM Rbfdot Estaciones	GBM Estaciones
1	23	0	1	0
2	29	49	0	0
3	34	39	6	2
4	8	10	12	3
5	3	0	24	4
6	1	0	22	6
7	0	0	11	6
8	0	0	9	5
9	0	0	5	6
10	0	0	2	7
11	0	0	1	6
12	0	0	5	9
13	0	0	0	12
14	0	0	0	7
15	0	0	0	15
16	0	0	0	10

Tabla 12. Estaciones con mejor resultado por puntos cercanos.

Como vemos en la tabla, con LM los mejores resultados se obtienen utilizando entre 1 y 6 puntos cercanos. La mayor parte de las estaciones están concentradas en 1, 2 y 3 puntos, siendo un total de 86 de 98 las estaciones que obtienen así su mejor resultado.

Fijándonos ahora en la columna de SVM con kernel polinómico, los resultados están todavía más concentrados, obteniendo todas las estaciones su mejor resultado utilizando los 2, 3 y 4 puntos más cercanos.

Pasamos ahora a SVM con kernel Gaussiano. Los mejores resultados se empiezan a dispersar. La mayoría de estaciones (78 en total) obtienen su mejor resultado utilizando entre 4 y 8 puntos cercanos. De las 20 estaciones restantes, 7 obtienen su mejor resultado con menos puntos y

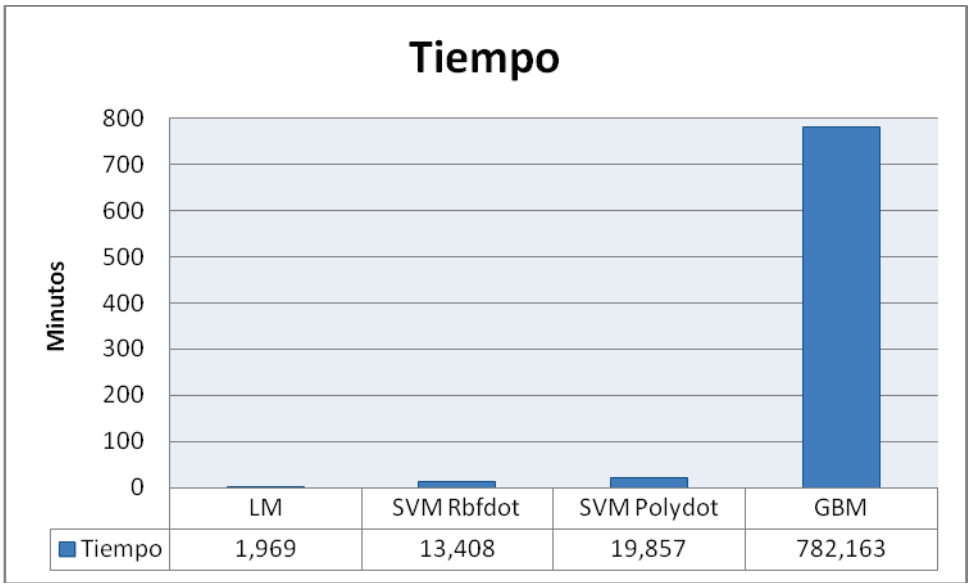
el resto con más, sin subir en ningún caso de los 12 puntos más cercanos a la estación.

Por último, con GBM los resultados se dispersan aún más. Las estaciones obtienen sus mejores resultados utilizando desde los 3 puntos más cercanos hasta los 16. La mayor concentración se da al utilizar entre los 12 y los 16 puntos más cercanos (53 estaciones en total). El resto de modelos (entrenados utilizando desde 3 hasta 11 puntos cercanos) son algo menos eficaces, ya que obtienen el mejor resultado en pocos casos (entre 2 y 7 estaciones cada uno).

4.3.5. Comparación de los tiempos de ejecución de cada método.

En este punto, compararemos los tiempos de ejecución de cada método para tener en cuenta la rapidez de los métodos, además de su precisión, al elegir el método más apropiado para utilizar en el problema de las predicciones a corto plazo de la acumulación de radiación solar.

En la Gráfica 29 podemos ver el tiempo medio, medido en minutos, que ha tardado cada método en ejecutar todas las pruebas pertenecientes al estudio de una estación.



Gráfica 29. Tiempo medio de ejecución por estación.

Como podemos ver en la gráfica, el tiempo de ejecución de LM es el más bajo, ya que no llega a 2 minutos de media por estación. Hay que recordar que el estudio completo de una estación, que es lo que se mide en estos tiempos, se compone de 16 modelos y 32 predicciones (un

modelo por cada cantidad de puntos cercanos a la estación posible y dos predicciones por cada modelo, una de entrenamiento y otra de test).

Los dos casos de SVM tienen tiempos medios que se pueden considerar cercanos. Utilizando kernel Gaussiano, el tiempo medio es de unos 13 minutos y medio, mientras que al utilizar el kernel polinómico, el tiempo medio asciende a casi 20 minutos. Teniendo en cuenta que con kernel polinómico el modelo tarda más en aprender que con kernel Gaussiano y las predicciones son menos precisas, en SVM podemos coronar al kernel Gaussiano como kernel más efectivo, ya que los modelos aprenden más rápido y mejor.

Por último, el tiempo de ejecución para el último método utilizado (GBM) se dispara muy considerablemente con respecto a los demás métodos. Este es el método más preciso, por lo que al menos el incremento de tiempo de ejecución sirve para obtener una mayor precisión en los resultados, pero si comparamos los resultados de GBM con los de SVM con kernel Gaussiano, no hay mucha diferencia entre los errores de un método y otro, por lo que dependiendo de los recursos disponibles para abordar el problema, podría merecer más la pena sacrificar esa poca precisión que obtenemos de más con GBM a cambio de ahorrar recursos y tiempo en las predicciones con SVM Rbfdot.

Como ya he comentado, los tiempos de la gráfica anterior abarcan todas las pruebas del estudio completo de una estación. A continuación, vamos a comparar los tiempos de una sola prueba por estación, que es la situación más cercana a la real, pues por cada estación sólo se lanzaría una ejecución con un número de puntos cercanos concreto y un método. En esta comparación, para cada método utilizaremos el número de puntos cercanos que de media obtiene el error más bajo.

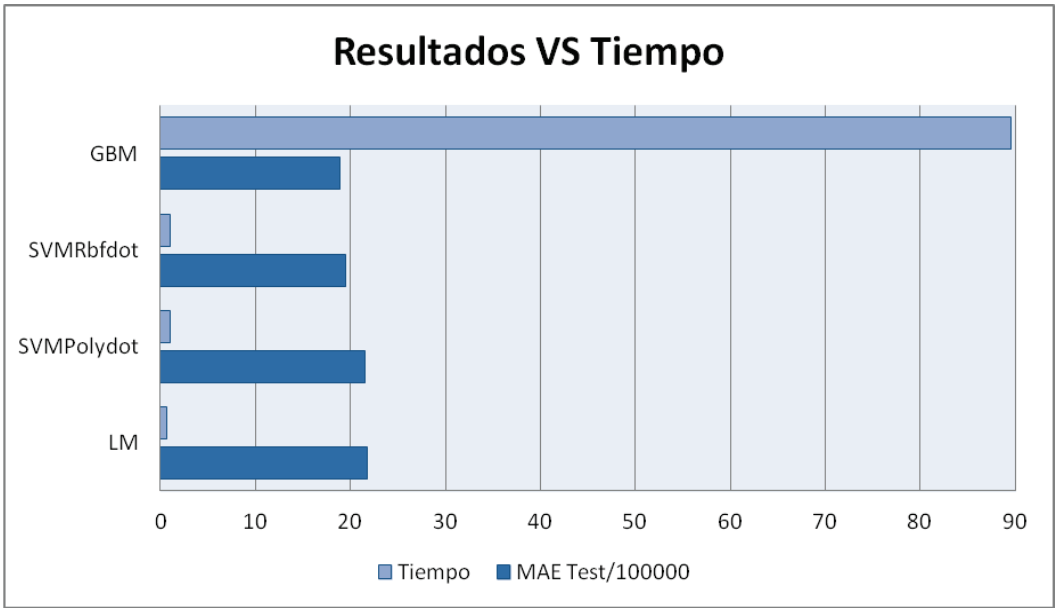
Si consultamos de nuevo la Tabla 10 del apartado 4.3.2, vemos que LM y SVM Polydot obtienen el menor MAE de test promedio con 2 puntos cercanos, SVM Rbfdot lo obtiene con 5 puntos y GBM con 15. Compararemos los resultados y el tiempo de ejecución de un modelo por método con los puntos cercanos correspondientes a cada uno. Esto lo haremos solamente con la estación 1, a modo de comprobación. De este modo, podremos ver el tiempo que tardaría cada método en ejecutar el mejor de sus casos para una misma estación.

La siguiente tabla muestra los resultados de esta prueba. Los tiempos están medidos en minutos e incluyen la lectura de los datos de entrada, la creación del modelo y las predicciones de entrenamiento y test.

MÉTODO	MAE TRAIN	MAE TEST	TIEMPO (Min.)
LM	2116753	2180607	0,67
SVMPolydot	1994669	2157423	0,97
SVMRbfdot	1836192	1948855	1,05
GBM	1158658	1890822	89,58

Tabla 13. Resultados de la prueba con la mejor combinación de parámetros y nodos del GEFS de cada método para la Estación 1.

Como podemos ver en la tabla, GBM consigue el error de test más bajo necesitando casi 1 hora y media de ejecución. Sin embargo, SVM Rbfdot consigue un error bastante parecido en apenas 1 minuto. Esto refuerza nuestro razonamiento anterior con los tiempos de ejecución totales. GBM es el método más preciso y por consiguiente el más adecuado, pero en caso de que se pueda sacrificar algo de precisión a cambio de ahorrar en recursos y tiempo de cómputo, SVM Rbfdot sería una buena opción a tener en cuenta.



Gráfica 30. Representación del error frente al tiempo consumido.

Capítulo 5: Conclusiones y Trabajos Futuros.

En este capítulo, expondremos las conclusiones que hemos podido sacar a partir de los datos obtenidos en la experimentación, dando respuesta a las incógnitas que nos hemos ido planteando durante el desarrollo de este estudio. También propondremos ideas para posibles trabajos futuros que se podrían realizar tomando este trabajo como punto de partida, o mejoras que se podrían intentar llevar a cabo sobre lo que hemos hecho.

5.1. Conclusiones.

En este Trabajo de Fin de Grado, se ha llevado a cabo un estudio con diferentes técnicas de aprendizaje automático para abordar el problema de las predicciones a corto plazo de radiación solar acumulada, utilizando datos de 98 estaciones de la Mesonet de Oklahoma y datos procedentes del *NOAA/ESRL Global Ensemble Forecast System* (GEFS) para un mapa de 16x9 nodos. Se han empleado tres métodos de regresión diferentes (Regresión lineal, Máquinas de Vectores de Soporte y *Gradient Boosting Regression*) en un total de seis experimentos (LM, SVM Rbfdot, SVM Polydot, SVM Tanhdot, GBM y XGBoost). En cada uno de ellos, creamos diversos modelos para poder estudiar la influencia del número de puntos del GEFS en la precisión de las predicciones. Además, se ha hecho un estudio pormenorizado de los diferentes parámetros de cada método para ver la influencia de sus valores en la precisión de los modelos. Aunque ha habido dos experimentos no finalizados (SVM Tanhdot, cancelado durante la fase de ajuste de parámetros debido a los pésimos resultados, y XGBoost, que no se pudo terminar por los problemas expuestos en el apartado 4.2.6.2), esto no nos ha impedido cumplir los objetivos planteados al inicio del proyecto.

Según muestran los resultados de los experimentos, los métodos de aprendizaje no lineales parecen ser mejores para este problema que los métodos lineales, ya que obtienen errores menores. El mejor método de todos los probados es GBM porque es el que nos proporciona los errores más bajos. SVM Rbfdot obtiene errores muy similares a GBM cuando usamos pocos puntos del GEFS, pero cuando el número de puntos es grande, se produce un ligero sobreaprendizaje. No obstante, si nos fijamos en los tiempos de ejecución de ambos métodos además de en los resultados, GBM necesita mucho más tiempo para entrenar sus modelos que SVM Rbfdot, y como ya se ha comentado, cuando utilizamos pocos puntos del GEFS como datos de entrada, ambos métodos predicen de manera muy similar. Por ello, puede haber casos en los que merezca la pena utilizar SVM Rbfdot en lugar de GBM, ya que se ahorrarían recursos y tiempo de cómputo a cambio de sacrificar un poco de precisión.

En cuanto a cómo afecta el número de puntos del GEFS utilizados al error de predicción, la influencia cambia según el método utilizado. De media, el error en LM y SVM Polydot tiende a empeorar al incrementar el número de puntos del GEFS utilizados para entrenar los modelos, produciéndose un sobreaprendizaje que puede considerarse bastante alto. Por otro lado, SVM Rbfdot obtiene de media el error más alto cuando se utiliza el punto más cercano. Según se incrementa el número de puntos del GEFS utilizados hasta llegar a 5, el error tiende a mejorar. A partir de ahí, al utilizar más puntos, el error vuelve a aumentar. Por último, con GBM el error tiende a mejorar cuantos más puntos del GEFS se utilizan para realizar las predicciones. A partir de esto, podemos concluir que de los métodos utilizados, sólo GBM es capaz de aprovechar toda la información proporcionada por los 16 puntos del GEFS, aunque a partir de 6 o 7 puntos, la mejora es ya pequeña.

Hemos podido llegar a una conclusión sobre qué técnicas proporcionan las mejores predicciones de energía solar a corto plazo; hemos profundizado en el artículo del que partíamos, puesto que hemos tratado los datos con más detalle; hemos extendido dicho artículo, ya que hemos estudiado nuevos métodos para predecir, además de los que se utilizaron en él con parámetros e implementaciones diferentes; y hemos validado los resultados y conclusiones a las que se llegaron en él. De este modo, podemos dar por finalizado este Trabajo de Fin de Grado de manera satisfactoria, porque hemos conseguido sacar las conclusiones que buscábamos y hemos cumplido todos los objetivos que nos habíamos marcado.

5.2. Trabajos futuros.

En este apartado, hablaremos de algunos posibles trabajos que se podrían desarrollar a partir de este proyecto.

La propuesta más evidente es la de finalizar el experimento con XGBoost. Como ya quedó explicado en el apartado 4.2.6.2, dicho experimento no pudo completarse debido a un problema con el paquete que implementaba XGBoost en R. Sería muy interesante que se trabajase en solucionar este problema y poder finalizar el experimento, porque viendo los resultados que veníamos obteniendo en el estudio de parámetros y los tiempos de ejecución de dichas pruebas, podemos estimar que XGBoost necesitaría menos de la mitad del tiempo que necesita GBM para crear los modelos, y las predicciones podrían ser bastante cercanas a las de GBM en cuanto a precisión.

Otra posible tarea sería probar las Máquinas de Vectores de Soporte con kernel polinómico de grado mayor que 2. Nosotros probamos únicamente con grado 2 porque el objetivo era probar diferentes métodos y decidimos no probar con grados superiores por el tiempo que llevarían las ejecuciones.

Pero sería interesante ver cómo serían de precisas las predicciones con grado 3 o más.

También sería interesante estudiar cómo influye cada uno de los atributos de los puntos del GEFS en la precisión de los métodos. El GEFS recoge 15 variables meteorológicas en cada punto del mapa. En la Tabla 1 del apartado 3.2 se recogen dichas variables. Estudiando cómo influye cada una de ellas en la precisión de los métodos de predicción, se podrían descartar las que menos influyan para así optimizar el tiempo de entrenamiento de los modelos.

Por otro lado, dentro de los resultados de cada método, siempre hay algunas estaciones que presentan errores más altos que el resto de estaciones con un mismo método. Por poner un ejemplo, para la estación 37 todos los métodos que hemos utilizado obtienen errores considerablemente más altos que con otras estaciones. Las estaciones 60 y 88 son otros claros ejemplos de ello. Ya que no hay diferencias metodológicas en la recogida de datos ni en la experimentación con estas estaciones, todo apunta a que puede haber otro tipo de factores diferentes a los que hemos manejado en el estudio, como podrían ser factores geográficos de las estaciones o de los nodos, que afectan a la precisión de las predicciones, porque ocurre con todos los métodos. Se podría investigar esto para determinar por qué se produce, y seguramente sería de ayuda para el problema que hemos abordado.

La última propuesta que haré tiene relación con los datos de entrada utilizados en el estudio. Como ya expliqué en el apartado 3.2.1 de este documento, los datos de cada punto del GEFS que proporcionaba *Kaggle* se componían de 11 conjuntos de 75 atributos cada uno (las 15 variables meteorológicas x 5 mediciones al día). Dado el volumen de datos que eso suponía, nosotros utilizamos para el estudio la media de esos 11 conjuntos, y así ganar tiempo de cómputo para poder estudiar más técnicas de predicción. Ahora que hemos estudiado las diferentes técnicas y hemos llegado a la conclusión de cuál es la mejor, podría ser interesante realizar las pruebas con todos los datos disponibles y ver cómo afecta eso a los resultados. Como hacer esto con todos los métodos llevaría mucho tiempo, se podría hacer únicamente con GBM, que es el más preciso, o con las SVM Rbfdot, que hemos visto que están casi a la altura de GBM en cuanto a precisión y son mucho más rápidas creando los modelos. Esto serviría para ver cómo afectó a la precisión de los métodos la reducción de los datos de entrada que realizamos al crear los conjuntos de datos de entrenamiento y test para nuestros experimentos.

Capítulo 6: Planificación y Presupuesto.

En este penúltimo capítulo de nuestro trabajo, expondremos de la manera más detallada posible la planificación que hemos seguido en este proyecto y el presupuesto necesario para la realización del mismo.

6.1. Planificación.

Aquí, expondremos la planificación que hemos seguido para realizar las diferentes tareas que han compuesto este Trabajo de Fin de Grado.

Todas las tareas realizadas las hemos agrupado en las siguientes fases para facilitar la planificación:

- Recopilación y búsqueda de información previa: Esta fase incluye todas las tareas realizadas antes de iniciar la experimentación. Abarca toda la búsqueda de información sobre energía solar, recopilación de información sobre los diferentes modelos de predicción, herramientas utilizadas en el proyecto, etc.
- Experimentos: En ella, incluimos todas las tareas realizadas para cada uno de los diferentes experimentos que han compuesto este proyecto. Incluye la elección de los parámetros utilizados, el ajuste de los valores de dichos parámetros, la realización del código necesario para realizar las diferentes pruebas y su ejecución, la recopilación, tratamiento y evaluación de los resultados, la medición de los tiempos de ejecución, la resolución de los problemas surgidos durante la experimentación, etc. Tendremos una fase "Experimento" por cada uno de los experimentos explicados en el Capítulo 4 de esta memoria.
- Documentación: Esta última fase es la que incluye la redacción de esta memoria. Aunque la información recogida en este documento se ha ido recopilando a lo largo de todo el proyecto y algunas partes se han documentado durante el desarrollo para que no se olvidaran detalles importantes, la redacción final se ha realizado en esta fase.

A continuación, mostramos en la Tabla 14 cómo se han distribuido estas fases. Por cada fase se incluye el ciclo en el que se ha realizado, la fecha de inicio y fin, y los días dedicados a dicho ciclo. Se debe tener en cuenta que entre las fechas de inicio y fin de cada ciclo, se ha trabajado en el proyecto únicamente los días laborables. Y por regla general, cada uno de esos días se ha trabajado 3 horas. Por necesidades del proyecto, algunos días se le ha dedicado más o menos tiempo a las tareas realizadas, pero de media se han trabajado 3 horas por día.

FASE DEL PROYECTO	CICLO	FECHA INICIO	FECHA FIN	DÍAS
Recopilación y búsqueda de información previa	1	01/10/2014	16/10/2014	12
Experimento 1 (LM)	2	17/10/2014	30/10/2014	10
Experimento 2 (SVM Rbfdot)	3	31/10/2014	25/11/2014	17
Experimento 3 (SVM Polydot)	4	26/11/2014	17/12/2014	15
Experimento 4 (SVM Tanhdot)	5	18/12/2014	20/12/2014	2
Experimento 5 (GBM)	6	07/01/2015	15/03/2015	48
Experimento 6 (XGBoost)	7	16/03/2015	14/04/2015	19
Documentación	8	15/04/2015	20/05/2015	24

Tabla 14. Planificación del proyecto.

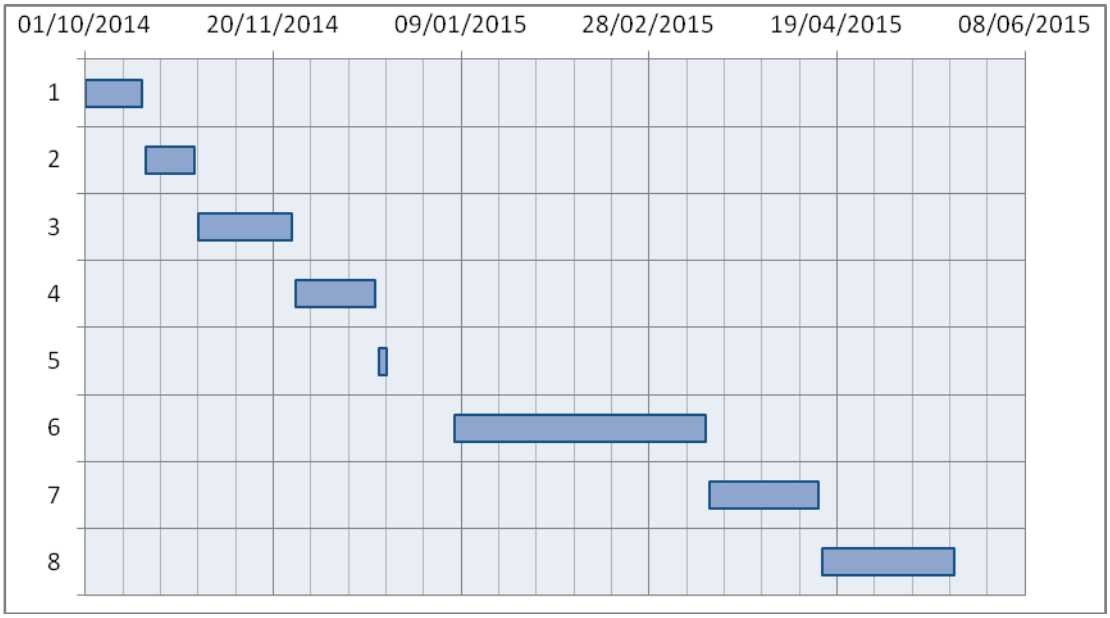


Ilustración 7. Diagrama de Gantt de la planificación del proyecto.

Esta planificación refleja cómo se ha desarrollado el proyecto, aunque inicialmente no estaba previsto que se alargase tanto en el tiempo. Esto fue inevitable debido a los elevados tiempos de cómputo necesarios para realizar las pruebas de algunos de los experimentos. Como hemos visto, algunos métodos de predicción tardaban mucho en crear sus modelos, lo que ralentizó considerablemente el avance del proyecto. Aunque se intentó compaginar las tareas de cada experimento en la medida de lo posible, siempre había que esperar a que terminasen las ejecuciones para realizar la mayoría de ellas.

El total de días dedicados al proyecto han sido 147. Teniendo en cuenta una media de 3 horas trabajadas por día, el tiempo de dedicación a este Trabajo de Fin de Grado han sido 441 horas. El TFG está cuantificado en 12 créditos ECTS. Cada crédito ECTS equivale a 25-30 horas de dedicación[48], lo que

supone un total de 300-360 horas de trabajo. El exceso de horas también se debe a la ralentización del trabajo por los tiempos de ejecución. Aunque se automatizaron las pruebas para que los equipos utilizados estuvieran continuamente ejecutando, incluso fuera de horas de trabajo, había que dedicarle tiempo a la monitorización de las ejecuciones para garantizar que todo seguía su curso de manera correcta y a la recopilación de los resultados.

6.2. Presupuesto.

En este apartado, detallaremos el presupuesto estimado para la realización de este proyecto. Dicho presupuesto se compone de tres tipos de gastos: los gastos de hardware, los gastos de software y los gastos de personal.

6.2.1. Gastos de hardware.

Para realizar este proyecto, han sido utilizados 3 equipos informáticos con las siguientes características técnicas:

- Procesador: Intel Core 2 Duo CPU E8400 @ 3.00GHz 3.00 GHz.
- Memoria RAM: 4,00GB.
- Disco Duro: 700GB.

Un equipo de estas características está valorado actualmente en 450€. En la siguiente tabla, se muestra el coste total detallado de todo el hardware utilizado.

HARDWARE	PRECIO + IVA	UNIDADES	COSTE
Equipos informáticos	450€	3	1350€
TOTAL			**1350€**

Tabla 15. Gastos de hardware del proyecto.

6.2.2. Gastos de software.

Respecto a la parte de software, los programas utilizados para la realización del proyecto son los siguientes:

- Sistema Operativo: Windows 7 Enterprise.

- Software: Microsoft Office 2010, R y RStudio.

En la Tabla 16, se refleja el coste de cada uno de los productos mencionados y el coste total del software utilizado.

SOFTWARE	VERSIÓN	PRECIO + IVA	UNIDADES	COSTE
Windows 7	Enterprise	*0€	1	0€
Microsoft Office	2013	69€	1	69€
R	3.1.1	0€	1	0€
R	3.1.2	0€	1	0€
RStudio	0.98.1062	0€	1	0€
TOTAL				**69€**

*Licencia incluida en el precio de los equipos.

Tabla 16. Gastos de software del proyecto.

6.2.3. Gastos de personal.

Para el cálculo de los gastos de personal, se tendrá en cuenta la planificación del apartado 6.1. Como ya detallamos en ese apartado, se necesitaron 147 días de trabajo para completar el proyecto, trabajando una media de 3 horas al día. El salario del ingeniero informático que hemos fijado es de 20€ brutos por hora de trabajo. Teniendo en cuenta estos datos, el coste de personal será el reflejado en la Tabla 17.

Nº INGENIEROS	DÍAS	HORAS/DÍA	PRECIO/HORA	COSTE
1	147	3	20€	8820€
TOTAL				**8820€**

Tabla 17. Gastos de personal del proyecto.

6.2.4. Gastos totales.

Por último, y teniendo en cuenta los gastos anteriores, el presupuesto total necesario para realizar este proyecto es el siguiente:

TIPO DE GASTO	COSTE
Gastos de hardware	1350€
Gastos de software	69€
Gastos de personal	8820€
TOTAL	**10239€**

Tabla 18. Gastos totales del proyecto.

Capítulo 7: Referencias.

Finalmente, en este último apartado listaremos todas las referencias, tanto electrónicas como bibliográficas, que hemos consultado para documentar el trabajo que hemos realizado de la forma más completa y precisa posible.

[1] Casas Úbeda, Gea López, Javaloyes Tarí, Martín Peña, Pérez Navarro, Triguero Sánchez, Vives Boix. (2007). *Educación Medioambiental: Las Energías Renovables*. Recuperado el 15 de Abril de 2015, de https://books.google.es/books?id=JDhoUfDmsvEC&pg=PA165&hl=es#v=onepage&q&f=false

[2] Osinergmin. (2013). *Introducción a las Energías Renovables*. Recuperado el 15 de Abril de 2015, de http://www2.osinerg.gob.pe/EnergiasRenovables/contenido/IntroduccionEnergiasRenovables.html

[3] Matras. *Desarrollo de un modelo operacional de predicción del recurso solar en escalas de horas a días***.** Recuperado el 15 de Abril de 2015, de http://matras.ujaen.es/es/solcasting.php

[4] Asociación de Empresas de Energías Renovables. (2014). *Estudio del Impacto Macroeconómico de las Energías Renovables en España en 2013*. Recuperado el 15 de Abril de 2015, de http://www.appa.es/descargas/Informe_2013_Web.pdf

[5] Greenpeace. (2008). *La energía solar puede dar electricidad limpia a más de 4.000 millones de personas para 2030*. Recuperado el 15 de Abril de 2015, de http://www.greenpeace.org/espana/es/news/2010/November/la-energ-a-solar-puede-dar-ele/

[6] Alberto García Cabrero. (2014). *La energía solar y el desarrollo del entorno socioeconómico*. Recuperado el 15 de Abril de 2015, de http://www.laenergiadelcambio.com/energia-solar-desarrollo-economico

[7] Kelsey Mowatt. *Social, Economic & Environmental Impacts of Renewable Energy Systems*. Recuperado el 15 de Abril de 2015, de http://www.ehow.com/info_8420936_social-impacts-renewable-energy-systems.html

[8] Kaggle. (2013). *AMS 2013-2014 Solar Energy Prediction Contest*. Recuperado el 01 de OCtubre de 2014, de https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest

[9] Kaggle. *Web Oficial de Kaggle*. https://www.kaggle.com/about

[10] Ocklahoma Mesonet. *Web Oficial de la Mesonet de Oklahoma*. https://www.mesonet.org/index.php/site/about

[11] Global Forecast System. *Web Oficial del NOAA/ESRL Global Ensemble Forecast System*. http://www.emc.ncep.noaa.gov/index.php?branch=GFS

[12] Aler, Ricardo; Martín, Ricardo; Valls, José M.; Galván, Inés M. (2015). *A Study of Machine Learning Techniques for Daily Solar Energy Forecasting using Numerical Weather Models*. Recuperado en Enero de 2015, de http://e-archivo.uc3m.es/handle/10016/19310

[13] GNU Operating System. (2007). *Web Oficial de la GNU General Public License*. http://www.gnu.org/copyleft/gpl.html

[14] Universidad Complutense de Madrid. *Análisis de Regresión Lineal.* Recuperado el 02 de Octubre de 2014, de http://www.ucm.es/info/socivmyt/paginas/profesorado/benitacompostela/tema2.doc

[15] Corinna Cortes y Vladimir Vapnik. (1995). *Support-Vector Networks*. Recuperado el 02 de Octubre de 2014, de http://link.springer.com/article/10.1007%2FBF00994018

[16] Jerome H. Friedman. (1999). *Greedy Function Approximation: A Gradient Boosting Machine*. Recuperado en Octubre de 2014, de http://statweb.stanford.edu/~jhf/ftp/trebst.pdf

[17] Jerome H. Friedman. (1999). *Stochastic Gradient Boosting*. Recuperado en Octubre de 2014, de https://statweb.stanford.edu/~jhf/ftp/stobst.pdf

[18] Javier Seoane, Carlos P. Carmona, Rocío Tarjuelo y Aimara Planillo. (2014). *Árboles con remuestreo: Random Forest y Boosted Regression Trees*. Recuperado en Octubre de 2014, de http://www.uam.es/personal_pdi/ciencias/jspinill/CFCUAM2014/RF_BRT-CFCUAM2014.html

[19] Kaggle. *Tools Used By Competitors*. Recuperado en Octubre de 2014, de https://www.kaggle.com/wiki/Software/history/33284

[20] R-Project. *Web Oficial del Entorno de Programación R*. http://www.r-project.org/

[21] R-Project. *R-Project Contributors*. Recuperado en Octubre de 2014, de http://www.r-project.org/contributors.html

[22] Mi Rincón Matemático. (2010). *Software Open Source*. Recuperado en Octubre de 2014, de http://www.mates.byethost4.com/software/software-open-source.html

[23] R-Project. *Repositorio Oficial de Paquetes de R*. http://www.cran.r-project.org/web/packages

[24] RStudio. *Web Oficial de RStudio*. http://www.rstudio.com/about

[25] Programacióndesarrollo.es. (2011). *Qué es un Entorno de Desarrollo Integrado (IDE)*. Recuperado en Abril de 2015, de http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide

[26] CPlusPlus. *Web Oficial del Lenguaje de Programación C++*. http://www.cplusplus.com

[27] Qt Framework. *Web Oficial del Marco Qt*. http://www.qt.io/qt-framework

[28] R Documentation. *Fitting Linnear Models.* Recuperado en Octubre de 2014, de https://stat.ethz.ch/R-manual/R-patched/library/stats/html/lm.html

[29] R Documentation. *Predict method for Linear Model Fits*. Recuperado en Octubre de 2014, de https://stat.ethz.ch/R-manual/R-patched/library/stats/html/predict.lm.html

[30] R-Project. *kernlab: Kernel-based Machine Learning Lab*. Recuperado en Octubre de 2014, de http://cran.r-project.org/web/packages/kernlab/index.html

[31] R Documentation. *ksvm: Support Vector Machines*. Recuperado en Noviembre de 2014, de http://www.rdocumentation.org/packages/kernlab/functions/ksvm

[32] John C. Platt. (2000). *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. Recuperado en Abril de 2015, de http://courses.cs.tau.ac.il/0368-4341/shared/Papers/SMO/smo-book%20+%20pseudocode.pdf

[33] R Documentation. *predict.ksvm: Predict Method for Support Vector Object*. Recuperado en Noviembre de 2014, de http://www.rdocumentation.org/packages/kernlab/functions/predict.ksvm

[34] R-Project. *gbm: Generalized Boosted Regression Models*. Recuperado en Enero de 2015, de http://cran.r-project.org/web/packages/gbm/index.html

[35] Yoav Freund y Robert E. Schapire. (1999). *A Short Introduction to Boosting*. Recuperado en Abril de 2015, de http://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf

[36] R Documentation. *gbm: Generalized Boosted Regression Modeling*. Recuperado en Enero de 2015, de http://www.rdocumentation.org/packages/gbm/functions/gbm

[37] R Documentation. predict.*gbm: Predict method for GBM Model Fits*. Recuperado en Enero de 2015, de http://www.rdocumentation.org/packages/gbm/functions/predict.gbm

[38] Github. *Official XGBoost Documentation*. Recuperada en Marzo de 2015, de https://github.com/dmlc/xgboost/blob/master/doc/README.md

[39] OpenMP. *Web Oficial de OpenMP*. http://openmp.org/wp/about-openmp

[40] Github. *Official XGBoost R-package Documentation*. Recuperada en Marzo de 2015, de https://github.com/dmlc/xgboost/blob/master/R-package/vignettes/xgboostPresentation.Rmd

[41] R Packages. *eXtreme Gradient Boosting (Tree) library*. Recuperado en Marzo de 2015, de http://rpackages.ianhowson.com/cran/xgboost/man/xgboost.html

[42] R Packages. *Predict method for eXtreme Gradient Boosting model*. Recuperado en Marzo de 2015, de http://rpackages.ianhowson.com/cran/xgboost/man/predict-xgb.Booster-method.html

[43] R-Project. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*. Recuperado en Marzo de 2015, de http://cran.r-project.org/web/packages/e1071/index.html

[44] Microsoft. *¿Qué es la Prevención de ejecución de datos?. Recuperado en Abril de 2015, de* http://windows.microsoft.com/es-es/windows-vista/what-is-data-execution-prevention

[45] Stackoverflow. *Web Oficial Stackoverflow*. http://stackoverflow.com/tour

[46] Stackoverflow. (2014). *R crashes randomly*. Recuperado en Abril de 2015, de http://stackoverflow.com/questions/21807874/r-crashes-randomly

[47] R-Project. *Repositorio Oficial de Paquetes de R: XGBoost*. Recuperado en Mayo de 2015, de http://cran.r-project.org/web/packages/xgboost/index.html

[48] BOE. (2003). *Ministerio de Educación, Cultura y Deporte*. Recuperado en Mayo de 2015, de http://www.boe.es/boe/dias/2003/09/18/pdfs/A34355-34356.pdf

Además de todas las referencias listadas anteriormente, también se consultaron los siguientes libros para estudiar, entender y documentar los diferentes métodos de predicción utilizados en el trabajo:

- Sierra Araujo, B.; (2006); *Aprendizaje Automático: Conceptos Básicos y Avanzados. Aspectos Prácticos Utilizando el Software Weka*; Madrid, España; Pearson Prentice Hall; ISBN 9788483223185.

- Hernández Orallo, J.; Ramírez Quintana, M. J.; Ferri Ramírez, C.; (2004); *Introducción a la Minería de Datos*; Madrid, España; Pearson Prentice Hall; ISBN 9788420540917.

- Castillo Ron, E; Cobo, A.; Gutiérrez J. M.; Pruneda, R. E.; (1999); *Introducción las Redes Funcionales con Aplicaciones: Un Nuevo Paradigma Neuronal*; Madrid, España; Paraninfo; ISBN 9788428325257.

- Schapire, R. E.; Freund, Y.; (2012); *Boosting: Foundations and Algorithms (Adaptive Computation and Machine Learning Series)*; United States of America; Massachusetts Institute of Technology; ISBN 9780262017183.

Anexo I: Competencia en Inglés.

I.I. Introduction.

Renewable energy is the one that is obtained from natural resources that doesn't run out, either because they are able to regenerate themselves by natural means or because of the immense amount of energy they contain. In addition, their environmental impact on the emission of greenhouse gases is zero. The most significant renewable energies are solar, wind, geothermal, hydroelectric, tidal, biofuels and biomass.

This type of energy has been a very important part of the energy used by humans since ancient times, especially wind, hydro and solar energy.

Since the 70s, renewable energies are considered an alternative to traditional energy. This was caused by their availability guaranteed, both at present and in the future (which is not the case of fossil fuels, which takes thousands of years to form themselves), and its lower environmental impact.

At present, renewable energy sources are gaining importance due to the enhanced greenhouse effect and consequent global warming.

Within renewable energies, one of the most prominent types is solar energy. This type of renewable energy is obtained from the electromagnetic radiation from the sun. Each year, the solar radiation brings to Earth the energy equivalent to several thousand times the amount of energy consumed by humans. Collecting solar radiation in the right way, it can be transformed into electrical or thermal energy by using solar panels.

The main problem is that solar energy is an intermittent and non-deterministic resource. This problem can cause the mains is unstable, for example, when clouds are passing or any other situation like that. Therefore, it is vital for us to be able to predict when it is going to happen a situation that can reduce solar radiation.

As the number of renewable plants integrated in the overall electricity grid grows, management and distribution of electricity are becoming more complex by the characteristic intermittence of this resource, thus putting at risk the stability of the system.

To address this problem, the development of solar radiation prediction systems is very important. This would help increase the amount of renewable energy plants without increasing the risk of power outages or surges, so an important development in this area is needed.

I.II. Objectives.

Given the problem of solar radiation predictions that we just introduced, the main objective of this project is to study and find the best statistical and machine learning techniques to make predictions for the production of solar energy.

We will start from the competition "Solar Energy 2013-2014 AMS Prediction Contest" celebrated between July and November 2013 in Kaggle, which is a web platform dedicated to data prediction competitions, where companies and researchers publish their data, and statisticians and data-miners around the world compete to produce the best models.

The aim of the competition was to discover which learning techniques provide the best short-term predictions of daily total solar energy acummulated in a total of 98 stations of the Oklahoma Mesonet, which served as solar plants in the competition. The Oklahoma Mesonet is a worldwide network of environmental monitoring stations designed to measure the size and duration of mesoscale weather phenomena. The term Mesonet is a combination of the english words "mesoscale" and "network". In meteorology, mesoscale phenomena are the phenomena that lasts between 1 and 12 hours, has an horizontal length of between 1 and 100 kilometers, or a height of between 1 and 10 kilometers.

Numerical weather prediction input data for the competition came from NOAA/ESRL Global Ensemble Forecast System (GEFS). These data included all 11 ensemble members and forecasts in five timesteps of the day: 12, 15, 18, 21 and 24 hours. Data for 1994 to 2007 were provided as training data, and data for 2008 and 2009 as a test data set. For the final evaluation of the competition, more recent and private test data were used.

In Illustration 8, it is represented the location of the points from which GEFS data comes in a grid of 16x9 points, and location of the 98 stations of the Oklahoma Mesonet, which as I said, were taken as solar plants in the competition. GEFS data grid points are represented in blue color, and solar plants or stations are represented in red color.

This issue was previously addressed in the article "A Study of Machine Learning Techniques for Solar Energy Daily using Numerical Weather Forecasting Models", published by both tutors of this end-of-degree project, Ricardo Aler and Joseph M. Valls, along with Ricardo Martin and Agnes M. Galvan, all of them are members of the Department of Informatics at University Carlos III of Madrid, in the area of Computer Science and Artificial Intelligence.

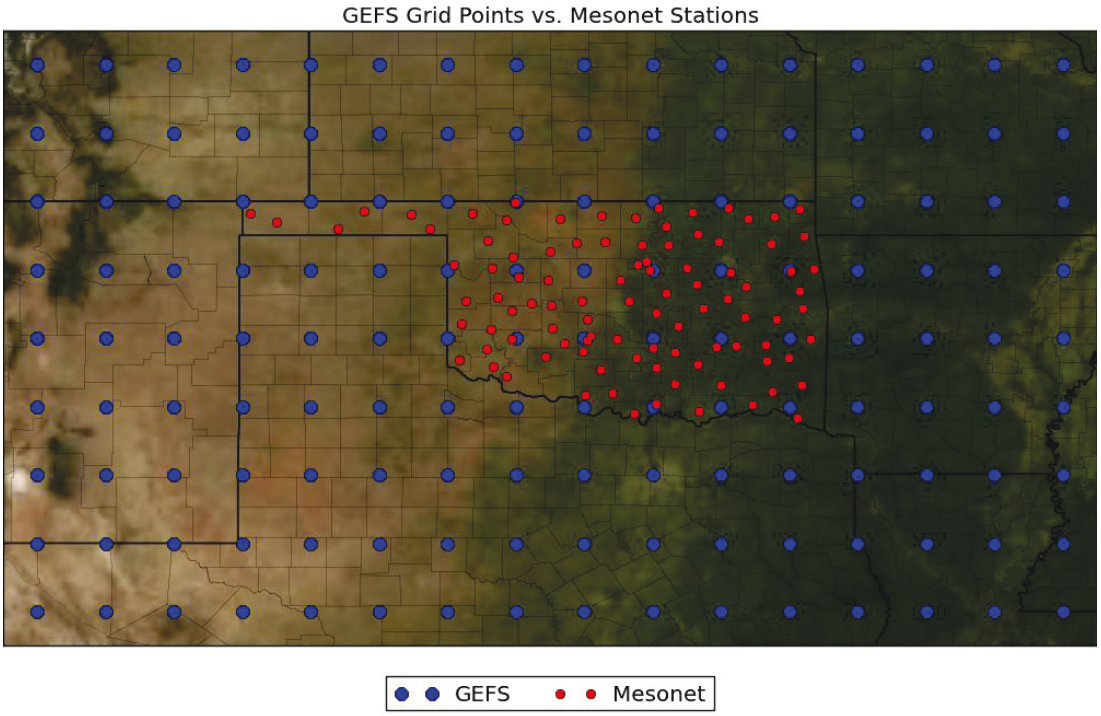


Illustration 8. Map with the grid points of GEFS data and the Oklahoma Mesonet stations. © 2013 Kaggle, all rights reserved.

With this starting point, the objective of this Final Degree Work is not only to study what are the best techniques for predictions of solar energy. This work will be a deepening of that article, and we will try and analyze the data in more detail, and also an extension of it, because we will use different methods and parameters used in the article. In addition, we validate the results of the article.

The main objective is to study how the prediction error changes by increasing the number of GEFS points used to make the prediction. Also, different regression methods will be compared (Linear Regression, Support Vector Machines (SVM) linear, polynomial and Gaussian, and Gradient Boosting Regression). It will be studied if the dependency of the error with respect to the number of GEFS points depends on the method and how. It will also be measured the execution time of each method, in order to determine the computational cost that improvements suppose. As a secondary result of the whole process of construction of the models, we will try to visualize how the accuracy of the different methods depends on their parameters, in order to understand the application of these methods.

I.III. Results.

Now, we are going to analyze and compare the results obtained in the experiments we have done.

I.III.I. Results of all the methods used in their best parameters.

In this section, we will analyze and compare the results of all methods with their best parameters. Since the total amount of data is very large (3136 results for each one of the four experiments), we calculated the average results of the 98 stations to work with it. We also calculated the standard deviation.

Table 19 shows the average of the results of the 98 stations for each method and for each number of GEFS nearby grid points used to create the model.

ARITHMETIC AVERAGE OF THE MAE OF THE 98 STATIONS FOR EACH CASE								
Grid Points	LM		SVM POLYDOT		SVM RBFDOT		GBM	
	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST
1	2289589,82	2226545,94	2321784,58	2263055,27	2196550,87	2080477,52	1433200,9	1997768,6
2	2205927,54	2196503,69	2050742,93	2141330,05	2085322,09	2012230,18	1361301,61	1966513,37
3	2155292,63	2199000,66	1886149,2	2142795,49	2016145,87	1984942,75	1324578,65	1950992,64
4	2123212,89	2225742,41	1764662,19	2181572,53	1964489,48	1976414,51	1301536,5	1944217,23
5	2085323,99	2241261,39	1641073,04	2222744,5	1907660,56	1965965,04	1284744,22	1934425,93
6	2060647,59	2274730	1543617,22	2286196,27	1865242,3	1967298,56	1266364,46	1932713,33
7	2034913,27	2307471,56	1453658,57	2355101,9	1824087,23	1969488,25	1255941,24	1930188,59
8	2011922,71	2340029,53	1372274,55	2427752,25	1785997,48	1973412,02	1246510,74	1929248,28
9	1992059,11	2374271,29	1298665,08	2502964,91	1748731,41	1979743,26	1238488,93	1927967,67
10	1967797,59	2411634,95	1229816,15	2578013,71	1713036,1	1986219,95	1231161,67	1926007,09
11	1945203,74	2445595,17	1167567,06	2638755,12	1677991,11	1991857,77	1224003,93	1924637,72
12	1923462,26	2475462,85	1113231,1	2701952,38	1645416,98	1997688,29	1219801,74	1924914,71
13	1907094,74	2518148,03	1064772,82	2774397,83	1614970,32	2006401,94	1214928,31	1925338,69
14	1879688,12	2563143,89	1021052,68	2834152,89	1584838,75	2014062,92	1209771,55	1923344,02
15	1857045,63	2607515,01	980007,136	2895190,27	1554511,15	2020792,79	1206580,92	1916697,38
16	1834355,54	2645270,15	944514,366	2954773,54	1525627,42	2029916,26	1203861,02	1924423,4

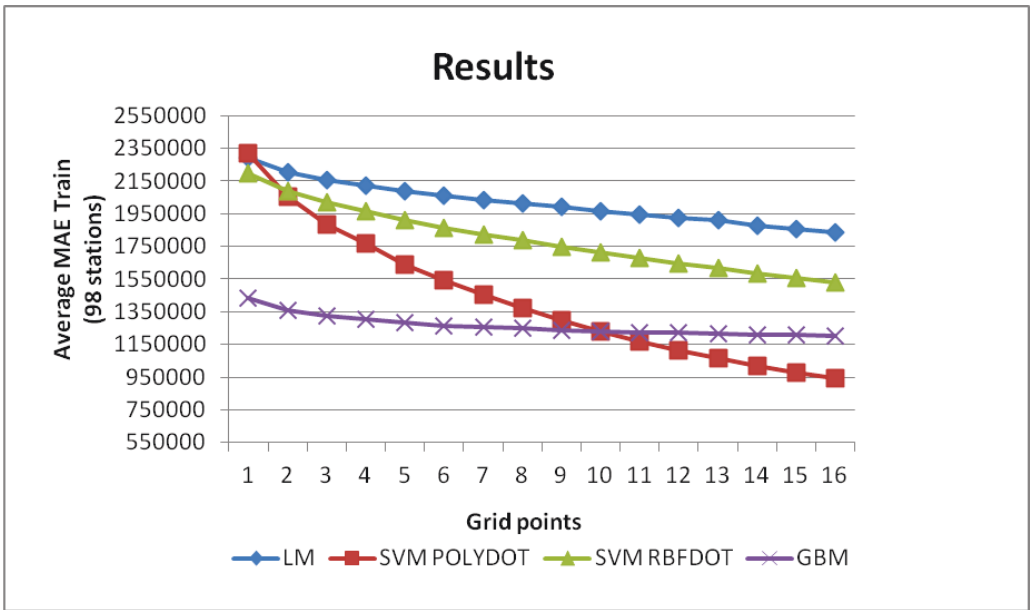
Table 19. Average results of the 98 stations for each method and every combination of nearby points used.

In the table, we can see that GBM is the most accurate method because it is the one that provides the lowest average MAE for test dataset. The Support Vector Machines with Gaussian kernel gives a little worse results than GBM but very similar.

Considering the average results, which are represented in this table, LM is better than SVM with polynomial kernel, since on average it obtains more accurate predictions in all cases except when we used between 2

and 5 grid points to train models. In those cases SVM are more accurate. However, if we consider the best results for each station, which were set out in paragraphs 4.2.1.4 and 4.2.3.4 (Tables 3 and 7 respectively), SVM with polynomial kernel obtains better results than LM in 77 of the 98 stations, about 78.5%, so by looking at this data, SVM Polydot would be better than LM. We could say that by using less input data, SVM with polynomial kernel is better prediction method than LM, but when more input data are used, LM is better to make predictions than SVM with polynomial kernel.

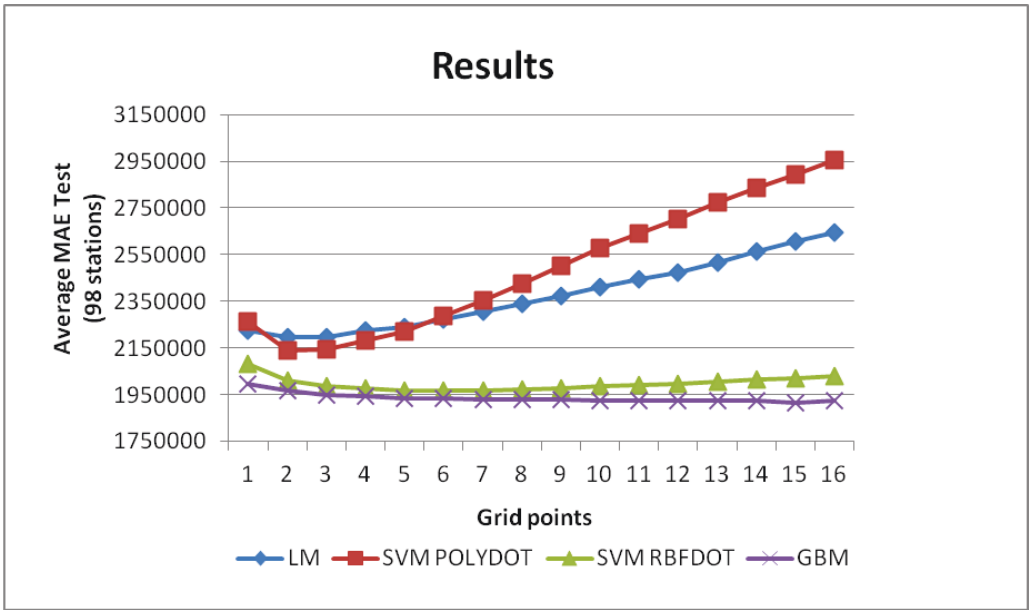
Now we will analyze the evolution of the training and test errors for the four methods.



Graphic 31. Evolution of train dataset MAE for each method.

In this graphic, it is represented the mean absolute errors obtained for training dataset for all prediction methods used, based on the amount of input data used (number of nearest GEFS points to each station used). The MAE represented on each curve is the average of the 98 stations.

We can see that in all methods, the mean absolute error of training data tends to decrease as the amount of input data is increasing. The explanation for this is that when more input data are used to train the models, it learns the data better, so the model predicts better with training datasets. This is not always good, it could happen that when new data are used, the model cannot make very accurate predictions because it has learned specific data and it cannot generalize.



Graphic 32. Evolution of test dataset MAE for each method.

In this other graphic, we represent the mean absolute errors obtained for test dataset by each predicting method, based on the amount of input data used (number of nearest points to each station used). The MAE represented on each curve is also the average of 98 stations, as in the previous case.

LM and SVM with polynomial kernel curves are quite similar. In them, we see that the results tend to worsen by increasing the amount of input data. Comparing both curves, we can also say that with less data, the linear model produces worse results than SVM with polynomial kernel, although when the input data volume is larger, the SVM predictions are far less accurate.

On the other hand, observing the curves of SVM with Gaussian kernel and GBM, they are also quite similar to each other. In the curve of SVM with Gaussian kernel, we see the error tends to improve slightly by increasing the amount of input data. The error varies rather little between points 5 and 8, and by continuing to increase the input data, the results begin to worsen. We could say that from that point on, overfitting occurs, since training error is getting lower but test error begins to increase (the model generalizes worse). In the GBM however, we see that the lowest error is reached with a lot of data (using input data from 15 of the 16 closest stations points), but the curve is fairly constant from point 5 because error barely varies. From point 15 to 16 the error gets worse, but the variation is very small.

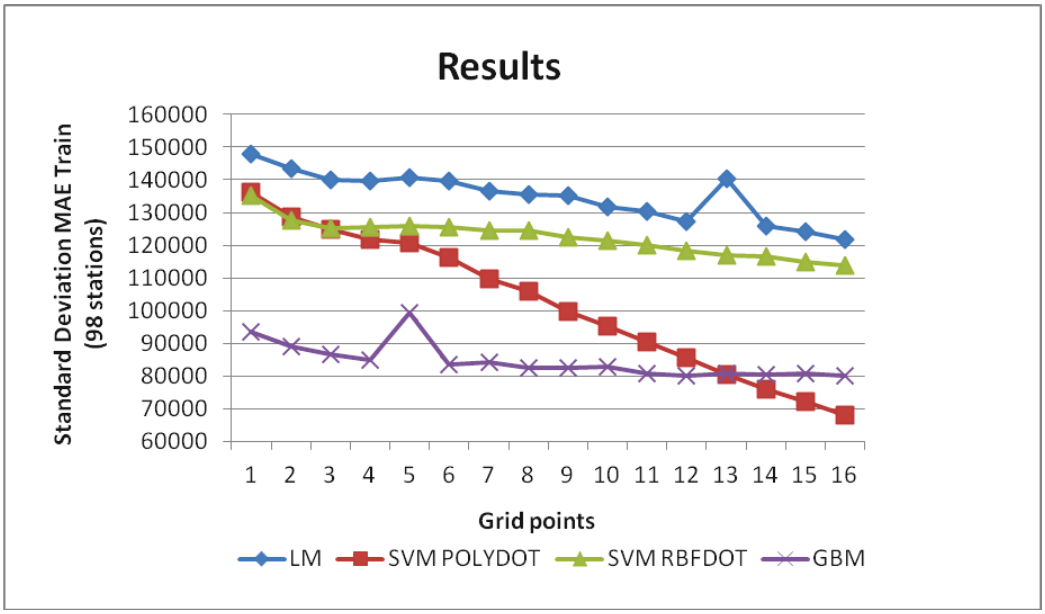
We obtain the best results with GBM, but as we could already see in the Graphic 28 (Variability of error per station with the 4 methods), the results of SVM with Gaussian kernel are very similar, and given the very different resources to execute a method and another, it could be considered more appropriate to use the Gaussian kernel SVM by sacrificing some accuracy in the results but saving resources and computation time.

STANDARD DEVIATION OF THE MAE OF THE 98 STATIONS FOR EACH CASE								
Grid points	LM		SVM POLYDOT		SVM RBFDOT		GBM	
	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST
1	147747,816	191195,343	136240,373	189734,613	135077,01	182517,635	93544,9262	191598,165
2	143481,653	185998,673	128731,21	186328,821	127692,286	185163,158	88910,3307	192354,274
3	140129,053	183654,079	124706,027	178081,357	125264,714	182674,188	86651,8845	188944,865
4	139704,977	181845,134	121783,579	180702,305	125353,68	183411,367	84996,4613	188651,235
5	140557,76	183041,809	120678,455	173848,428	125913,4	183593,334	99168,5863	188987,496
6	139525,072	182264,734	116317,958	173570,564	125627,261	183937,717	83528,7538	186296,423
7	136671,245	183862,695	109802,14	175996,384	124438,512	183755,857	84152,9097	185630,877
8	135345,08	181193,051	105887,382	179689,771	124363,173	182698,113	82577,5218	185347,118
9	135089,632	183772,658	99639,4744	185532,696	122551,908	183243,615	82438,2274	188143,882
10	131844,213	185980,448	95233,2961	182010,262	121294,28	183590,523	82924,3345	187275,409
11	130268,511	188261,008	90320,293	181591,442	119931,564	183614,036	80812,5625	187797,584
12	127368,182	189209,368	85605,0768	191261,024	118249,437	183271,53	80012,8128	187373,83
13	140437,778	192981,655	80285,4148	187597,508	116987,913	183034,196	80746,6948	188367,457
14	125710,282	195722,371	75850,1871	199577,57	116407,283	181832,552	80470,384	187232,487
15	124197,106	198021,804	71955,3011	200417,189	114952,213	180540,334	80574,3893	197575,631
16	121593,357	210604,908	68039,2739	210551,968	113811,44	179023,731	80048,1262	189987,148

Table 20. Standard deviation of the results of the 98 stations for each method and every combination of nearby points used.

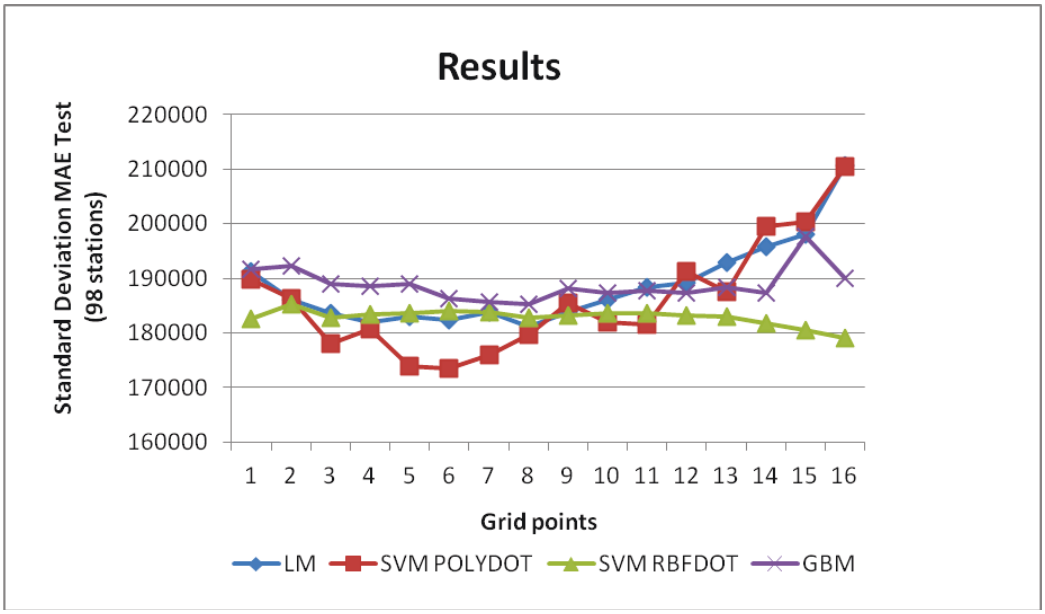
With the standard deviation we measure how much the specific values tend to move away from the average. In our study, the standard deviation for test dataset of the four methods moves between similar values. The slightest deviation occurs in Support Vector Machines with Gaussian kernel if using 1, 2, 9, 12, 13, 14, 15 and 16 nearby GEFS points, and polynomial kernel in the other models.

In Graphic 33 we can see the curves representing the standard deviation of training dataset for each method. We see that in all methods, the standard deviation tends to decrease when we increase the number of nearby GEFS points used to train the models.



Graphic 33. Standard deviation of train dataset MAE for each method.

In LM and GBM curves, a peak occurs at points 13 and 5 respectively. Given the behavior of the curves before and after these peaks, we can consider them as noise. Now, we represent the standard deviation of test dataset error in Graphic 34.



Graphic 34. Standard deviation of test dataset MAE for each method.

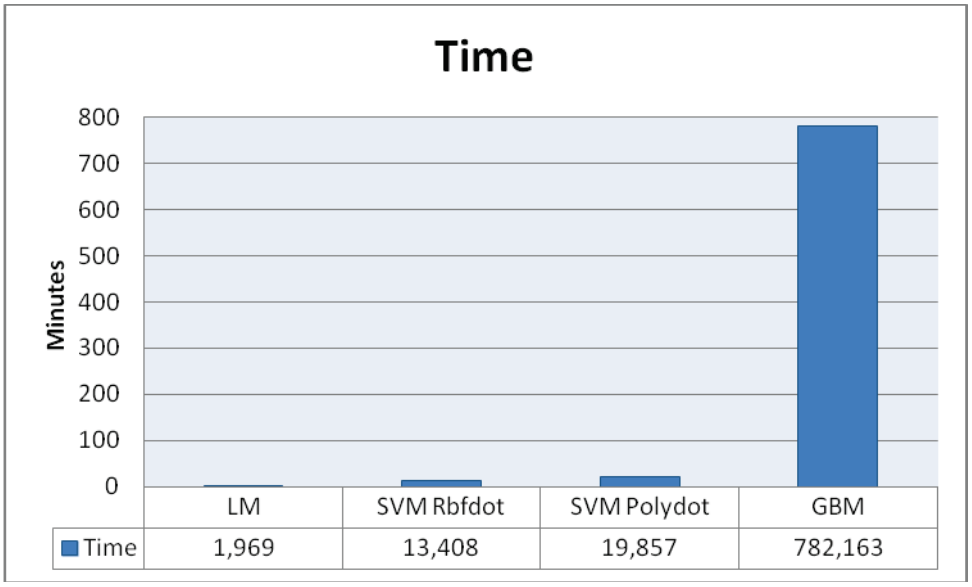
In this other graphic, we see that in LM and SVM Polydot, the points in the curves of standard deviation for test dataset are more dispersed than in the curves of GBM and SVM Rbfdot. That is because the variations

between models trained with different number of nearby grid points are more disparate.

I.III.II. Comparing runtimes of each method.

In this section, we will compare the runtimes of each method to take into account the speed of methods, in addition to its accuracy, to choose the most appropriate method to use in the problem of short-term predictions of solar radiation accumulation.

In Graphic 35 we can see the average time measured in minutes that every method needs to implement all the testing for the study of a station.



Graphic 35. Average runtime per station for each method.

As we can see in the graphic, the runtime of LM is the lowest, it is less than two minutes on average per station. It is necessary to remember that the complete study of a station, which is what is measured in these times, is composed of 16 models and 32 predictions (one model for each number of grid points near to the station and two predictions for each model, one with the training dataset and the other with the test dataset).

Both cases of SVM have average times that can be considered close to each other. Using Gaussian kernel, the average time is at about 13 and a half minutes, while using the polynomial kernel, the average time is nearly 20 minutes. Given that the model with polynomial kernel needs longer times to learn than the model with Gaussian kernel, and

predictions are less accurate, in SVM we can crown the Gaussian kernel as the most effective, since the models learn faster and better.

Finally, runtime for the last method used (GBM) shoots up very significantly from the other methods. This is the most accurate method, so at least the increase in runtime is used to obtain greater precision in the results, but if we compare the results of GBM with SVM with Gaussian kernel, there is not much difference between the errors of one method and the other, so depending on the resources available to address the problem, it might be worth it to sacrifice the little more accuracy that we get with GBM in exchange for saving resources and runtime with SVM Rbfdot predictions.

As I said before, times of the previous graphic cover all tests of the complete study of a station. Now, we will compare the times of one test per station, which is the closest to the real situation, because for each station it would be launched just one execution with a determined number of grid points near to the station and using one predicting method. In this comparison, for each method we will use the number of nearby GEFS points which gets the lowest error in average.

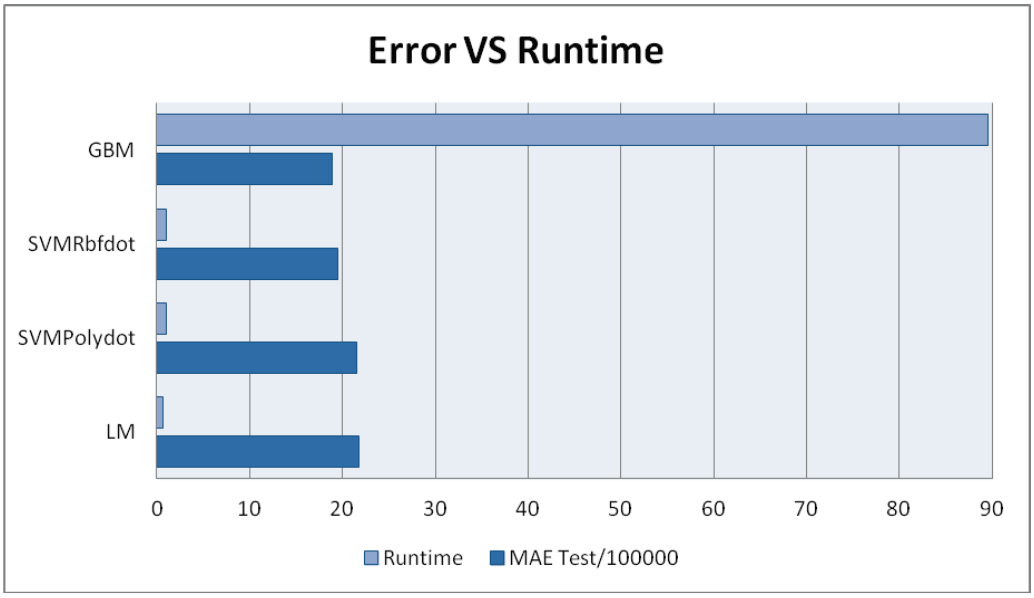
If we look back to Table 19 at chapter I.III.I, we see that LM and SVM Polydot get the lowest average test MAE with 2 nearby GEFS points, SVM Rbfdot gets it with 5 points and GBM with 15. We will compare the results and the runtime of a model per method with the corresponding nearby GEFS points for each one. We will do this with the station 1 only, as a test mode. Thus, we can see the time it would take to each method to run the best of their cases for the same station.

The following table shows the results of this test. Times are measured in minutes and they include reading the input data, creating the model and make predictions with training and test datasets.

METHOD	MAE TRAIN	MAE TEST	RUNTIME (Min.)
LM	2116753	2180607	0,67
SVMPolydot	1994669	2157423	0,97
SVMRbfdot	1836192	1948855	1,05
GBM	1158658	1890822	89,58

Table 21. Test results with the best combination of parameters and grid nodes of each method for Station 1.

As we can see in the table, GBM gets the lowest test error by consuming at about 1 hour and a half of runtime. However, SVM Rbfdot gets a very similar error in just 1 minute. This reinforces our earlier reasoning with total runtime. GBM is the most accurate and therefore best suitable method, but if you can sacrifice some accuracy in exchange for saving resources and runtime, SVM Rbfdot would be a good option to consider.



Graphic 36. Representation of error versus time consumed.

I.IV. Conclusions.

In this end-of-degree project, it has been conducted a study with different machine learning techniques to address the problem of short-term predictions of solar radiation acummulated, using data from 98 stations of the Oklahoma Mesonet and data from the NOAA/ESRL Global Ensemble Forecast System (GEFS) for a 16x9 grid. We have used three different regression methods (Linear Regression Model, Support Vector Machines and Gradient Boosting Regression) in a total of six experiments (LM, SVM Rbfdot, Polydot SVM, SVM Tanhdot, GBM and XGBoost). In each one of them, we created different models to study the influence of the number of grid nodes on the accuracy of predictions. Moreover, we made a detailed study of the different parameters of each method to see the influence of their values on the accuracy of the models. Although there have been two unfinished experiments (SVM Tanhdot, canceled during the parameter setting due to poor results, and XGBoost, that could not be completed because of the problems explained in section 4.2.6.2), this has not prevented us to reach the objectives raised in the project.

According to the results of experiments, nonlinear learning methods seem to be better for this problem than linear methods because they get lower errors. The best method of all is GBM because it is the one that provides the lowest error. SVM Rbfdot gets similar errors to GBM when we use few GEFS points, but when the number of points is large, a slight overlearning occurs. However, if you look at the runtimes of both methods in addition to the results, GBM takes more time to train their models than SVM Rbfdot, and as we already mentioned, when we use few GEFS points as input data, both methods predict very similarly. Therefore, there may be cases where it is worth using SVM Rbfdot instead of GBM, because resources and computing time would be saved in return for sacrificing some precision.

As for how the number of GEFS points used affects to prediction error, the influence is different depending on the method used. On average, the error in LM and SVM Polydot tends to worsen by increasing the number of GEFS points used to train the models, occurring an overfitting that can be considered quite high. Furthermore, SVM Rbfdot obtains the highest average error when the closest point is used. The error tends to improve as we increase the number of points used until we use the 5 closest GEFS points. From there, as we use more points, the error increases again. Finally, GBM error tends to improve, the more GEFS points are used to make predictions. From this we can conclude that of all the methods used, GBM is the only one that is able to take benefit of all the information provided by the 16 GEFS points, although from 6 or 7 points, the improvement is already small.

We were able to reach a conclusion about which techniques provide the best short-term predictions of solar energy; we have deepened in the article we used as a starting point, as we have processed the data in more detail; we have extended that article, since we have studied new predicting methods, in addition to the methods that were used in the article, but we have used them with different parameters and implementations; and we have validated the results and conclusions reached on it. Thus, we can finish this end-of-degree project satisfactorily, because we have managed to draw the conclusions we wanted and we have fulfilled all the objectives that we had set.